UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria Electrònica Industrial i Automàtica

DETECCIÓ DE PATRONS UTILITZANT TRACTAMENT DIGITAL D'IMATGES

Memòria i Annexes

Autor: Joan Ignasi de Mena i Obiols

Director: Pablo Buenestado Caballero

Co-Director: Leonardo Acho Zuppa

Convocatòria: Juny 2018

Resum

En el camp del reconeixement de caràcters es diferencien dos tipologies principals. La més estesa en aplicacions reals és el reconeixement òptic de caràcters escrits per una màquina (i per extensió, el de paraules) que es coneix com OCR (*Optical character recognition*). Es basa en un procés en què els elements a identificar tenen unes característiques morfològiques determinades i conegudes per l'aplicació que s'utilitza per fer el reconeixement. En aquesta tipologia s'inclouen aplicacions com la lectura de matrícules dels vehicles que accedeixen a un aparcament o la comprovació d'un carnet d'identitat en l'accés a un edifici.

Per altra banda, existeix el reconeixement intel·ligent de caràcters, conegut com ICR (*Intelligent character recognition*) que es diferencia de l'OCR per tenir com a objectiu el reconeixement de caràcters escrits a mà en qualsevol dels formats possibles. Aquesta tipologia d'identificació va acompanyada d'un procés d'aprenentatge i classificació de patrons conegut com a *Machine Learning* en què s'empren models estadístics per anar obtenint una millora contínua sense necessitat de modificar la programació utilitzada pel reconeixement.

És en aquest darrer terreny en què s'emmarca el present treball de fi de grau, per presentar un estudi de tècniques estadístiques de reconeixement de caràcters basat en la utilització de patrons. L'objectiu del treball es explicar el fonament d'aquestes tècniques i mostrar un exemple de com es poden dur al terreny pràctic mitjançant la utilització de l'eina informàtica *Matlab*. En concret, el treball s'ha centrat en la classificació de caràcters emprant el mètode *Fisher's linear discriminant analysis (FDA)* basat en els estudis de l'estadístic anglès Ronald Fischer (Londres 1890 – Adelaide 1962).

El treball descriu breument la base matemàtica en què es sustenta el mètode FDA i, finalment, proposa una rutina per identificar caràcters numèrics escrits a mà que obté uns resultats correctes en el 90% dels casos. Cal mencionar que existeixen altres tècniques estadístiques diferents a FDA que poden variar percentualment els resultats obtinguts com, per exemple, *Cross Validation* que pot millorar els resultats fins a un 6% addicional. Però en aquest treball hem estimat més interessant aprofundir en el mètode FDA.

La conclusió més interessant és que els resultats del reconeixement dels caràcters milloren substancialment quan més patrons tenim en la base de dades, la qual cosa suggereix que es podria assolir un alt percentatge d'encerts si el mateix sistema anés alimentant la base de patrons a mesura que s'anessin analitzant imatges. Per altra banda, com veurem en el cos de la memòria, la preparació de la imatge en quan a enquadrament, dimensions, nitidesa i traç del caràcter és fonamental per poder fer la comparativa de forma correcta. Seria doncs molt interessant que, de cara a una aplicació pràctica, en posterior treballs s'afegissin rutines complexes de tractament de la imatge que donessin un traç més tipogràfic i més nitidesa als caràcters escrits a mà.

Resumen

En el campo del reconocimiento de caracteres se diferencian dos tipologías principales. La más extendida en aplicaciones reales es el reconocimiento óptico de caracteres escritos por una máquina (y por extensión, el de palabras) que se conoce como OCR (*Optical character recognition*). Se basa en un proceso en el que los elementos para identificar tienen unas características morfológicas determinadas y conocidas por la aplicación que se utiliza para hacer el reconocimiento. En esta tipología se incluyen aplicaciones como la lectura de matrículas de los vehículos que acceden a un aparcamiento o la comprobación de un carné de identidad en el acceso a un edificio.

Por otra parte, existe el reconocimiento inteligente de caracteres, conocido como ICR (*Intelligent character recognition*) que se diferencia del OCR por tener como objetivo el reconocimiento de caracteres escritos a mano en cualquiera de los formatos posibles. Esta tipología de identificación va acompañada de un proceso de aprendizaje y clasificación de patrones que es conocido como *Machine Learning* en el cual se emplean modelos estadísticos para ir obteniendo una mejora continua sin necesidad de modificar la programación utilizada para el reconocimiento.

Es en este último terreno en el que se enmarca el presente trabajo de fin de grado, para presentar un estudio de técnicas estadísticas de reconocimiento de caracteres basado en la utilización de patrones. El objetivo del trabajo es explicar el fundamento de estas técnicas y mostrar un ejemplo de cómo se pueden llevar al terreno práctico mediante la utilización de la herramienta informática *Matlab*. En concreto, el trabajo se ha centrado en la clasificación de caracteres utilizando el método *Fisher's lineal discriminant analysis* (FDA) basado en los estudios del estadístico inglés Ronald Fischer (Londres 1890 - Adelaida 1962).

El trabajo describe brevemente la base matemática en que se sustenta el método FDA y, finalmente, propone una rutina para identificar caracteres numéricos escritos a mano que obtiene unos resultados correctos en el 90% de los casos. Cabe mencionar que existen otras técnicas estadísticas diferentes a FDA que pueden variar porcentualmente los resultados obtenidos como, por ejemplo, *Cross Validation* que puede mejorar los resultados hasta un 6% adicional, pero en este trabajo hemos estimado más interesante profundizar en el método FDA.

La conclusión más interesante es que los resultados del reconocimiento de los caracteres mejoran sustancialmente cuantos más patrones tenemos en la base de datos, lo que sugiere que se podría alcanzar un alto porcentaje de aciertos si el mismo sistema fuera alimentando la base de patrones a medida que se fueran analizando imágenes. Por otra parte, como veremos en el cuerpo de la memoria, la preparación de la imagen en cuanto a encuadre, dimensión, nitidez y trazo del carácter que contiene es fundamental para poder hacer la comparativa de forma correcta. Sería pues muy interesante que, de cara a una aplicación práctica, en posteriores trabajos se añadieran rutinas complejas de tratamiento de la imagen que dieran un trazo más tipográfico y más nitidez a los caracteres escritos a mano.

Abstract

In the field of character recognition, two main types are distinguished. The most widespread in real applications is the optical recognition of characters written by a machine (and by extension, of words) that is known as OCR (Optical character recognition). It is based on a process in which the elements to identify have certain morphological characteristics and are known by the application used to make the recognition. This typology includes applications such as the reading of license plates of vehicles that access a parking or the verification of an identity card at the entrance to a building.

On the other hand, there is intelligent recognition of characters, known as ICR (Intelligent character recognition) that differs from OCR by having as objective the recognition of handwritten characters, in any of the possible formats. This type of identification is accompanied by a process of learning and classification of patterns that is known as Machine Learning in which statistical models are used to obtain continuous improvement without having to modify the programming used for recognition.

It is in this last area in which the present End-of-Degree work is enclosed, to present a study of statistical techniques of character recognition based on the use of patterns. The objective of the work is to explain the basis of these techniques and show an example of how they can be carried to the practical field by using the *Matlab* computer tool. In particular, the work has focused on the classification of characters using the Fisher's linear discriminant analysis (FDA) method based on the studies of the English statistician Ronald Fischer (London 1890 - Adelaide 1962).

The work describes briefly the mathematical basis on which the FDA method is based and, finally, proposes a routine to identify numerical characters written by hand that obtain correct results in 90% of the cases. It is worth mentioning that there are other statistical techniques different to FDA that can vary percentage of the results obtained, such as Cross Validation that can improve the results up to an additional 6%, but we have considered more interesting to deepen the FDA method.

The most interesting conclusion is that the results of the recognition of the characters improve substantially the more patterns we have in the database, which suggests that a high percentage of successes could be reached if the same system were feeding the base of patterns as they were analysing images. On the other hand, as we will see in the body of this memory, the preparation of the image in terms of framing, dimension, sharpness and trace of the character that it contains is fundamental to be able to make the comparison in a correct way. It would be very interesting, therefore, for the practical application of subsequent works to add complex image processing routines that would give a more typographical stroke and more clearness to handwritten characters.

Índex

1. INTRODUCCIÓ

En el camp del reconeixement de caràcters es diferencien dos tipologies principals. La més estesa en aplicacions reals és el reconeixement òptic de caràcters escrits per una màquina (i per extensió, el de paraules) que es coneix com OCR (*Optical character recognition*). Es basa en un procés en què els elements a identificar tenen unes característiques morfològiques determinades i conegudes per l'aplicació que s'utilitza per fer el reconeixement. En aquesta tipologia s'inclouen aplicacions com la lectura de matrícules dels vehicles que accedeixen a un aparcament o la comprovació d'un carnet d'identitat en l'accés a un edifici.

Per altra banda, existeix el reconeixement intel·ligent de caràcters, conegut com ICR (*Intelligent character recognition*) que es diferencia de l'OCR per tenir com a objectiu el reconeixement de caràcters escrits a mà en qualsevol dels formats possibles. Aquesta tipologia d'identificació va acompanyada d'un procés d'aprenentatge i classificació de patrons conegut com a *Machine Learning* en què s'empren models estadístics per anar obtenint una millora contínua sense necessitat de modificar la programació utilitzada pel reconeixement.

És en aquest darrer terreny en què s'emmarca el present treball de fi de grau que presenta un estudi d'una tècnica matemàtica de reconeixement de dígits basada en la utilització de patrons.

La motivació per a fer un treball d'aquestes característiques és el repte que suposa l'aplicació de tècniques ICR i de funcions matemàtiques avançades per assolir una funcionalitat de reconeixement de text escrit a mà que, habitualment, s'associa al camp OCR, i es posa com a fita assolir un alt percentatge de reconeixements positius.

L'abast inicial del treball és el reconeixement individualitzat de dígits numèrics escrits a mà tot i que, finalment, s'ha ampliat amb la funcionalitat de separació de dígits que permet el reconeixement dígits que pertanyen a grups numèrics.

En quant a l'estructura del treball, aquest es divideix en una part teòrica i una part pràctica. Primer es descriuen tècniques matemàtiques per a la detecció de contorns i binarització d'imatges que són emprades en la part pràctica per a la manipulació de les imatges. A continuació s'explica la base matemàtica en què es sustenta el mètode de classificació emprat, *Fisher's linear discriminant analysis (FDA),* basat en els estudis de l'estadístic anglès Ronald Fischer (Londres 1890 – Adelaide 1962). Entrant en el terreny més pràctic, s'estudien les capacitats bàsiques de gestió d'informació del programari Matlab, fent èmfasi en les que permeten la manipulació d'imatges. Finalment es proposa una rutina per identificar caràcters numèrics escrits a mà que obté uns resultats correctes en el 90% dels casos. Aquest resultat fa que els objectius del projecte es considerin assolits.

Cal mencionar que existeixen altres tècniques estadístiques diferents que poden variar percentualment els resultats obtinguts com, per exemple, *Cross Validation* que pot millorar els resultats fins a un 6%. Però en aquest treball s'ha estimat més interessant aprofundir en el mètode FDA.

2. Part teòrica

2.1. Adquisició de la Base teòrica

La primera activitat sorgida de la reunió d'inici del projecte va ser la lectura del document sobre processament digital d'imatges [1]. En aquest document s'exposen aplicacions de les derivades parcials en el processament d'imatges, de les que cal destacar el Laplacià i el Diferenciador.

2.2. El Laplacià

El Laplacià, eina que es pot explicar como una derivada doble de la imatge (involucra segones derivades parcials), es pot utilitzar per a millorar la qualitat d'imatges una mica borroses, ressaltant les intensitats de la imatge en la part més discontínua i atenuant-les en les parts més contínues.

$$f(x_0, y_0) = f(x_0 + 1, y_0) + f(x_0 - 1, y_0) - 4f(x_0, y_0) + f(x_0, y_0 + 1) + f(x_0, y_0 - 1)$$

Equació 1. El Laplacià.

La representació gràfica de l'equació del Laplacià és la següent:

	$f(x_0, y_0 + 1)$	
$f(x_0 - 1, y_0)$	$f(x_0, y_0)$	$f(x_0 + 1, y_0)$
	$f(x_0, y_0 - 1)$	

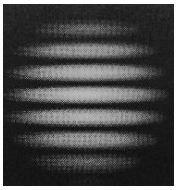
Imatge 1. Representació gràfica del Laplacià

Portant l'equació al terreny pràctic, el Laplacià es representa en forma d'un filtre espacial en què per cada píxel tractat es multiplica el seu valor per una constant determinada (-4 en l'exemple de la imatge 2) i se l'hi suma el valor de multiplicar una segona constant. Realitzant aquesta operació a tots els píxels de la imatge obtenim una imatge més nítida.

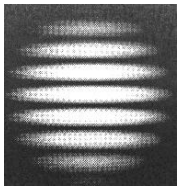
0	1	0
1	-4	1
0	1	0

Imatge 2. Valors numèrics a aplicar en el Laplacià

Veiem un cas pràctic. Aplicant a la imatge 3 l'algoritme del Laplacià obtenim la imatge 4.



Imatge 3. Imatge original [1]



Imatge 4. Imatge filtrada

El script de *Matlab* utilitzat es troba en l'Annex 1 – Script Laplacià.

2.3. El Diferenciador

El Diferenciador és una eina que s'utilitza per a detectar contorns d'imatges mitjançant l'ús d'una aproximació de derivades que detecta canvis bruscos en la pendent.

$$f'(x_0) = \frac{f(x_0 + \Delta) - f(x_0)}{\Delta}$$

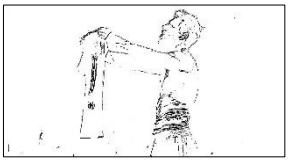
Equació 2. El Diferenciador com una aproximació de derivada.

El Diferenciador s'aplica realitzant un recorregut en les dues diagonals de la imatge per restar a cada píxel el valor de l'anterior, i posteriorment s'aplica un llindar per crear la imatge binària en la qual es destaca el contorn.

A continuació es mostra un cas pràctic.

Aplicant a la imatge 4 l'algoritme de *Matlab* del Diferenciador obtenim la imatge 5.

Imatge 5. Imatge original [2]



Imatge 6. Imatge després d'aplicar el Diferenciador

El script de *Matlab* corresponent a la funció de diferenciador es troba en l'Annex 2 – Script Diferenciador.

La lectura i realització de les activitats proposades en el document han estat molt útils per iniciar el treball amb més coneixement de fonaments matemàtics relacionats amb el tractament d'imatges abans d'aprofundir en l'ús del programari *Matlab*.

2.4. Statistical Machine Learning

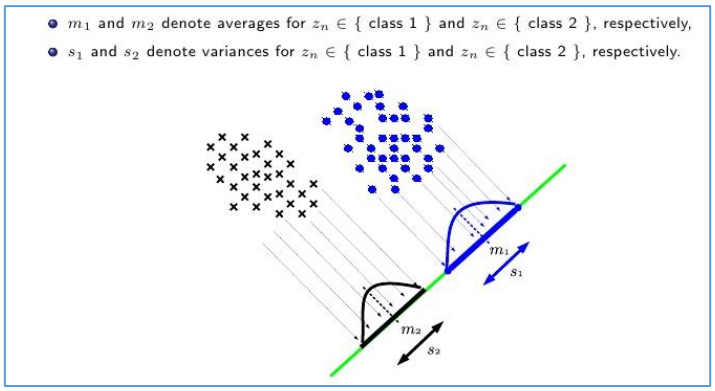
Després de realitzar la part teòrica bàsica es va procedir a analitzar el treball de Masashi Sugiyama[1] titulat *An Introduction to Statistical Machine Learning* [3], que descriu els tipus d'aprenentatge que pot realitzar una màquina, conceptes matemàtics relacionats amb la distribució de probabilitat (expectativa, mitjana, desviació estàndard, ...), diferències entre elements per probabilitat, estimació estadística i reconeixement de patrons per models estimatius.

Per el present treball ens vam centrar en les tipologies d'Aprenentatge de màquina que es classifiquen en tres categories:

- *Supervised Learning*: Aquest mètode es basa en què l'aprenent, la computadora, adquireix els coneixements del supervisor (l'usuari de la computadora) mitjançant preguntes que reben les corresponents respostes. Exemples d'àmbit d'aplicació són el reconeixement de text escrit o la predicció meteorològica. En aquest mètode parlarem de tres categories:
    - Regressió: si les respostes del supervisor són valors reals. Per exemple una mida.
    - Classificació: si la resposta és categòrica. Per exemple: Sí o no.
    - Rang: si la resposta és un valor ordinal del tipus bo, normal o dolent.
- *Unsupervised Learning*: És l'aprenentatge en què no hi ha supervisor i la màquina aprèn per si mateixa. Un exemple d'aplicació podria ser la detecció de dades atípiques en un bloc d'informació o la classificació d'informació.
- *Reinforcement Learning*: És similar al *Supervised Learning*, però amb la particularitat que el supervisor no contesta preguntes a la màquina, sinó que avalua el seu comportament i li dóna informació perquè millori la seva resposta. S'aplica a alguns jocs d'ordinador i per a control de robots autònoms.

En l'àmbit del present treball aplica el concepte de *Supervised Learning* donat que l'objectiu que perseguim és utilitzar patrons validats per un supervisor per alimentar la base de dades en què farem la comparativa per a classificar els elements analitzats.

El capítol de l'estudi de Masashi Sugiyama més interessant per al present treball és el relatiu a *Fisher's linear discriminant analysis* utilitzat per al reconeixement de dígits escrits a mà. Aquest treball s'ha inspirat en aquest mètode per a realitzar l'script de reconeixement de dígits.



Imatge 7. Fischer's linear discriminant analysis [7]

[1] Doctor en Enginyeria del Tokyo Institute of Technology

2.5. Fisher's linear discriminant analysis

En aquest apartat es descriu el fonament matemàtic del mètode *Fisher's linear discriminant analysis* (des d'ara FDA) basat en els estudis del estadístic anglès Ronald Fischer (Londres 1890 – Adelaide 1962). FDA és la implementació que Fischer va fer de Linear Discriminant Analysis (LDA), una tècnica d'aprenentatge supervisat per a la classificació de dades.

El primer objectiu que busca FDA és aconseguir que la separabilitat entre les diferents classes dels elements matricials que volem classificar sigui la més gran possible, facilitant així la classificació. Es tracta, doncs, de projectar les dades de les matrius en un espai unidimensional. Cal primer obtenir una matriu de dispersió dins de cada categoria i una matriu de dispersió entre totes les categories. FDA proposa maximitzar el quocient entre les dues matrius per aconseguir una projecció que ens permeti discriminar més entre cada una de les classes. La segona part important del mètode consisteix en disposar de models de dades en un espai que anomenarem Patró, que tingui una dimensió menor o igual que les dades que voldrem contrastar contra ell. Aquest espai Patró ha de ser alimentat amb dades etiquetades en el que s'anomena procés d'aprenentatge i que farà que disposem d'una funció de densitat de cada una de les classes, és a dir sabrem el comportament probable dels valors dels components de cada una de les classes que formen l'univers analitzat. Una cop tenim aquests patrons classificarem la mostra en aquella classe que tingui el valor del terme de classe a posteriori més elevat.

Veiem el desenvolupament matemàtic de tot el que hem comentat. Com volem buscar la màxima separabilitat entre les classes que composen el nostre univers de mostra, maximitzarem una funció objectiu en què en el numerador estigui la matriu de dispersió entre classes i en el denominador la matriu de dispersió dins de cada classe.

Sent $x_1 \ldots x_n$ els patrons de "d" dimensions i "c" el número de classes de que està format el nostre univers a analitzar, i assumint que cada classe té N patrons crearem un vector per cadascun dels N patrons de cadascuna de les "c" classes, la qual cosa ens proporcionarà una projecció unidimensional de cada patró i en permetrà obtenir la major separabilitat entre ells.

L'element a classificar també el convertirem en un vector per a poder comparar elements de dimensions afins. El següent pas serà calcular la covariància mitjana S de tots els patrons de totes les classes, assumint que la covariància es comuna totes elles i, finalment, obtindrem la mitjana m_i de cadascun dels elements que composen cada vector.

Classificarem la mostra en aquella classe en què el terme de decisió per la classe a posteriori sigui més alt. Aquest terme de decisió de la classe a posteriori el calcularem aplicant la fórmula:

$$Td = x^T \hat{\mathrm{S}}^{-1} m_i - \frac{1}{2} m_i^T \hat{\mathrm{S}}^{-1} m_i$$

Equació 3. Terme de decisió de la classe posterior.

On:

- $\hat{S}^{-1}$ és la inversa de la matriu de covariància mitjana de tots els elements patró.
- m_i és la mitjana de cada una de les classes dels patrons
- m_i^T és la mitjana de totes les classes
- x^T és l'element que volem classificar

3. Part pràctica

3.1. *Matlab* com a eina de tractament d'imatges

La segona part del treball es va focalitzar en adquirir coneixements i pràctica en l'ús de *Matlab*, per realitzar diferents tractaments de la imatge, interactuar amb l'usuari i treballar amb matrius en què s'emmagatzemen les imatges. Es va consultar diferents document a Internet [4].

3.1.1. El programari *Matlab*

Ens centrarem primer de tot en aspectes generals del programari *Matlab*, creat per la companyia MathWorks (Natick, Massachusets, Estats Units d'Amèrica) al 1985. Aquest programari es pot utilitzar en els entorns *Unix*, *Windows*, *Mac* i *Linux*. Està escrit en llenguatge *C*, *C++* i *Java*, i pot cridar funcions escrites en *C* o en *Fortran*. La versió actual estable és la R2018a, de 15 de Març de 2018. Com molts altres programaris, *Matlab* proveeix llicències gratuïtes pel món acadèmic.

Imatge 8. Icona de *Matlab* [8]

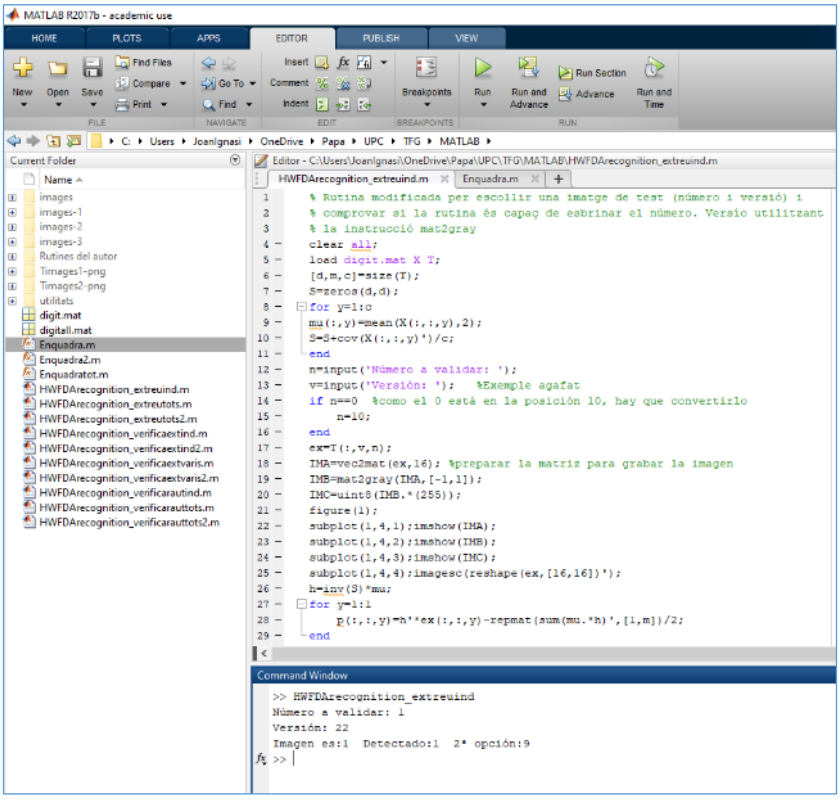
Les principals prestacions de *Matlab* són la manipulació de matrius, la representació de dades i funcions, la implementació d'algoritmes, la creació d'interfícies d'usuari (GUI) i la comunicació amb programes en altres llenguatges i amb altres dispositius maquinari. Les capacitats del programari es poden ampliar amb les anomenades Caixes d'Eines (*Toolboxes*), entre les que cal destacar *Image Acquisition Tool*, per la gestió d'imatges, *Simulink,* eina molt coneguda en el món acadèmic per a la generació avançada de models*,* o *Guide*, per la creació de interfícies gràfiques d'usuari.

La base de funcionament de *Matlab* és l'script, que és on definim la seqüencia d'instruccions que volem executar per assolir un determinat objectiu. El llenguatge de programació és interpretat, la qual cosa permet executar el codi mitjançant tan en un script com en l'entorn interactiu, mitjançant l'escriptura de sentències en la finestra de comandes.

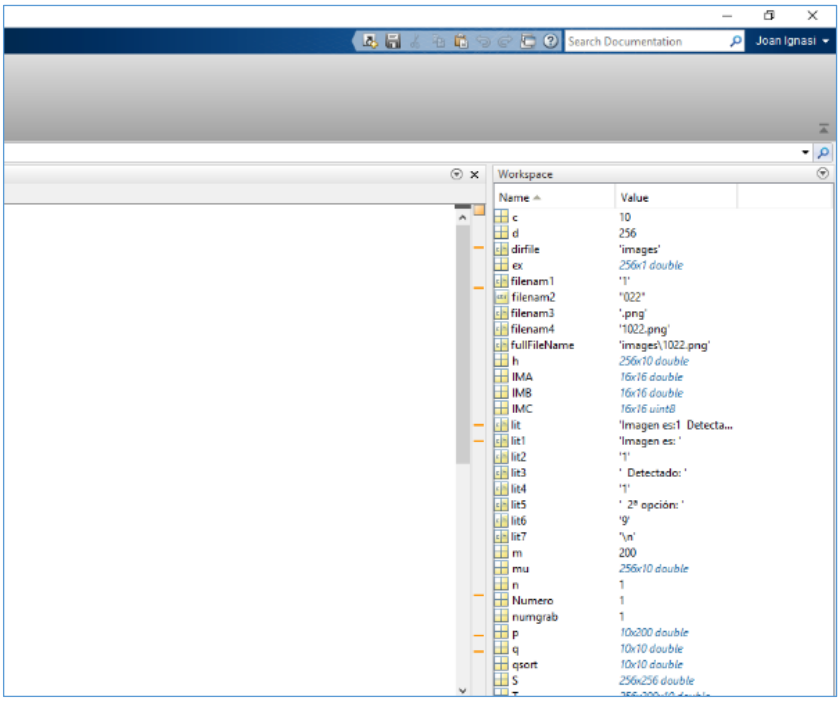
L'entorn de treball per defecte permet visualitzar 5 zones:

- Comandes i menús, característics de qualsevol programa en entorn Windows.
- La zona on trobem els diferents scripts realitzats i resta d'arxius auxiliars.
- La finestra de comandes (*Command Window*), zona en què interactuem amb el programa i on rebem els missatges del sistema relatius a errors, advertiments i presentació de dades.
- El *Workspace*, on el sistema ens presenta totes les variables, i les seves característiques, que estan intervenint en l'execució d'un script. Una gran funcionalitat de *Matlab* és precisament la facilitat amb què podem analitzar les dades que contenen les variables.
- L'*Editor*, que està associat amb l'edició d'un script i on s'enumeren seqüencialment les sentències que configuren l'script.

En les següent imatges podem veure l'aspecte de l'escriptori de *Matlab*: a la part superior la zona de menús, a l'esquerra la zona d'scripts, a la part central superior l'editor, a la part central inferior la finestra de comandes, i a la part dreta el *Workspace*.



Imatge 9. Part esquerra de l'escriptori de *Matlab*



Imatge 10. Part dreta de l'escriptori de *Matlab*

3.1.2. Tractament de dades

A *Matlab* les dades es gestionen mitjançant variables, com a la majoria d'entorns de programació, amb la particularitat que sempre són matrius. Podem declarar una variable especificant un tipus i una dimensió, però de fet no cal fer-ho, ja que amb una sentència d'assignació la variable queda declarada. Si, per exemple, volem crear una matriu de resultats de 3 partits d'un esport qualsevol entre 2 equips, i li volem donar dimensió 3x2 (3 files i 2 columnes) escriurem en l'script (o en la línia de comandes de la finestra de comandes) el següent:

```
RES=[2 6; 0 5; 1 3]
```

Amb aquesta simple comanda tindrem una matriu que representada visualment serà:

2	6
0	5
1	3

3.1.3. Operacions amb matrius

Podem fer operacions amb les matrius referint-nos simplement al nom de la variable que les referencia, i podem tractar les matrius en la seva totalitat, o bé gestionar informació d'un element, de files o de columnes senceres.

Exemples de comandes:

- Sumar un valor a tots els elements de la matriu RES:

Sentència	Resultat
RES = RES + 2	4 8 2 7 3 5

- Conèixer les dimensions de la matriu

Sentència	Resultat
[f,c]=size(RES)	f = 3 c = 2

- Invertir la matriu:

Sentència	Resultat
RESINV = RES'	4 2 3 8 7 5

- Conèixer el valor d'un element de la matriu (fila, columna)

Sentència	Resultat
RESINV(2,3)	5

- Conèixer el valor d'una fila de la matriu

Sentència	Resultat
RESINV(2,:)	8 7 5

- Conèixer el valor d'una columna de la matriu

Sentència	Resultat
RESINV(:,3)	3 5

- Assignar un valor a un element determinat de la matriu

Sentència	Resultat
RESINV(2,2)=9	4 2 3 8 9 5

- Assignar un valor a tota una fila de la matriu

Sentència	Resultat
RESINV(2,:)=0	4 2 3 0 0 0

- Assignar un valor a tota una columna de la matriu

Sentència	Resultat
RESINV(:,3)=7	4 2 7 0 0 7

- Multiplicar dues matrius

Sentència	Resultat
RES*RESINV	16 8 84 8 4 63 12 6 56

- Valor màxim que es pot trobar dins de la matriu RESI=[1 2 2; 3 3 2]

Sentència	Resultat
max(max(RESI))	3

- Mitjana de tots els valors de la matriu RESI=[1 2 2; 3 3 2]

Sentència	Resultat
mean(mean(RESI))	2.1667

- Ordenar les files d'una matriu M=[1 3 2; 7 9 8; 4 6 5]

Sentència	Resultat
qs=sort(M,1,'descend');	7 9 8 4 6 5 1 3 2
qs=sort(M,1,'ascend');	1 3 2 4 6 5 7 9 8

- Ordenar les columnes d'una matriu M=[1 3 2; 7 9 8; 4 6 5]

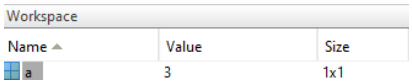
Sentència	Resultat
qs=sort(M,2,'descend');	3 2 1 9 8 7 6 5 4
qs=sort(M,2,'ascend');	1 2 3 7 8 9 4 5 6

Hi ha moltes més comandes addicionals, entre les que podem trobar-ne, entre d'altres, de trigonomètriques, logarítmiques o exponencials, però aquest recull seria massa extens i considerem que amb els grups de sentències exposats el lector es pot fer una primera idea de la potència que ens proveeix *Matlab* per la gestió i operació de dades.

3.1.4. Variables simples

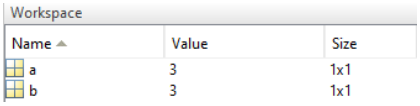
Basant-nos en aquest tractament de les variables com a matrius de diferents dimensions, tindrem que una variable simple en què vulguem emmagatzemar, per exemple, el valor d'un comptador serà considerada per *Matlab* com una matriu de dimensions 1x1.

- La comanda a=3, genera una matriu 1x1 amb el valor 3.



Imatge 11. La variable "a" en el *Workspace* de *Matlab*

- La comanda b=a, genera una matriu 1x1 amb el valor 3.
  És interessant aquesta propietat de *Matlab* que fa que una variable es crea de forma automàtica amb les característiques de la variable de la que es copia les dades, tal com es pot observar en la imatge 12.



Imatge 12. Les variables "a" i "b" en el *Workspace* de *Matlab*

3.1.5. Operacions amb vectors

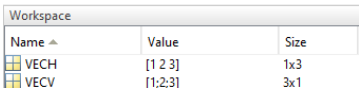
Com hem dit abans, *Matlab* considera totes les variables com matrius, per tant, si volem definir un vector de n posicions, el que ens caldrà fer és declarar una matriu nx1 o 1xn.

- Crear vector "horitzontal" 1x3:

Sentència	Resultat
VECH=[1 2 3]	1 2 3

- Crear vector "vertical" 3x1:

Sentència	Resultat
VECV=[1; 2; 3]	1 2 3



Imatge 13. Els vectors en el *Workspace*

Podem també convertir una matriu en un vector i viceversa de forma senzilla.

- Convertir Matriu RES=[1 2 3; 4 5 6; 7 8 9] a vector E

Sentència	Resultat
E=RES(:);	1 4 7 2 5 8 3 6 9
F=RES(:)';	1 4 7 2 5 8 3 6 9

- Convertir vector F=[1 3 2 7 9 8 4 6 5] a matriu M de 3 columnes

Sentència	Resultat
M=vec2mat(F,3)	1 3 2 7 9 8 4 6 5

3.1.6. Operacions estadístiques

Matlab també incorpora una sèrie d'instruccions relacionades amb estadística.

- Covariància d'una matriu A = [5 0 3 7; 1 -5 7 3; 4 9 8 10];

Sentència	Resultat
S=cov(A)	4.3333 8.8333 -3.0000 5.6667 8.8333 50.3333 6.5000 24.1667 3.0000 6.5000 7.0000 1.0000 5.6667 24.1667 1.0000 12.3333

- Distància de Mahalanobis

X = mvnrnd([0;0],[1 .9;.9 1],100);

Y = [1 1;1 -1;-1 1;-1 -1];

Sentència	Resultat
d = mahal(Y,X)	0.6288 19.3520 21.1384 0.9404

3.1.7. Instruccions de decisió i gestió de bucles

Com la majoria de llenguatges *Matlab* incorpora sentències per la presa de decisions, com "if...else" o "case"

Sentència	Resultat

if A=1
  B=2;
else
  B=3;
end
	Suposant que A val 1, B valdrà 2

Per la gestió de bucles tenim "while..." o "for..".

Sentència	Resultat

b=0;
for a=1:10
  b=b+1;
end
	b=10

Un cop analitzades les principals sentències, passem a analitzar les específicament relacionades amb el tractament d'imatges.

3.1.8. Sentències *Matlab* pel tractament de la imatge

Entre les sentències de tractament d'imatges de *Matlab* es troben les que permeten fer binarització, canvis de mida, retall, conversió a gama de grisos, erosió i dilatació o l'adquisició d'imatges i dades de l'usuari.

En la Taula 1 es detallen les instruccions bàsiques de manipulació d'imatges de *Matlab.*

Taula 1. Instruccions bàsiques de tractament d'imatges de *Matlab*

Categoria	Instrucció	Descripció	Format
Gestió imatges	imread	Llegeix una imatge i la col·loca en una matriu de dimensions equivalent	A=imread (imatge.jpg)
	imwrite	Grava una imatge a partir d'una matriu	imwrite (B,imatge.jpg)
Presentació	subplot	Permet dividir una figura en zones para representar vàries imatges	subplot (files, divisions, posició)
	imshow	Mostra la imatge corresponent a una matriu	imshow(C)
	disp	Mostra el valor d'una variable en la línia de comandes	disp(varA)
	fprintf	Permet presentar variables i textos formatats	fprintf(literal)
Modificació imatges	imdilate	Expandeix amb píxels els píxels de les bores seguint el patró indicat	imdilate(A, patró)
	imerode	Suprimeix píxels de los bordes seguint el patró indicat	imerode(A, patró)
	rgb2gray	Converteix una imatge de 3 capes en una imatge de 1 capa de grisos	rgb2gray(A)
	im2bw	Converteix una imatge en binària en funció del llindar de grisos	im2bw(B, llindar)
	medfilt2	Mediana de cada píxel en base als valors dels píxels confrontats	medfilt2(A)
	imresize	Canvia les dimensiones d'una imatge a els valores especificats	imresize(D,[mida, mida])
	Imcrop	Retalla la imatge amb les mesures especificades	imcrop(A[xmin,ymin, Ample, alt])
Informació imatges	Size	Proporciona les dimensions de la matriu	[files, columnes]=size(A)
	impixel	Proporcional el valor del píxel seleccionat interactivament	impixel(A)
	bwselect	Permet seleccionar els objectes connectats al píxel seleccionat	Bwselect(tipus selecció)

A la Taula 2 es mostren alguns exemples il·lustratius de les instruccions:

Taula 2. Exemples d'instruccions de tractament d'imatges de *Matlab*

Instrucció	Imatge original	Script	Imatge final

imdilate		clear all;
		A=imread('image12.jpg','jpg');
		Agray=rgb2gray(A);
		Abin=Agray>128;
		SE=strel('square',8);
		Adilate=imdilate(Abin,SE);
		figure(1);
		subplot (1,3,1);imshow(A);
		subplot (1,3,2);imshow(Adilate);
			

imerode		clear all;
		A=imread('image12.jpg','jpg');
		Agray=rgb2gray(A);
		Abin=Agray>128;
		SE=strel('square',8);
		Aerosion=imerode(Abin,SE);
		figure(2);
		subplot (1,3,1);imshow(A);
		subplot (1,3,2);;imshow(Aerosion);
			

rgb2gray		clear all;
		A=imread('peppers.png','png');
		Agray=rgb2gray(A);
		subplot(1,2,1);imshow(A);
		subplot(1,2,2);imshow(Agray);
		imwrite (Agray,'peppersb.png');
			

Instrucció	Imatge original	Script	Imatge final

im2bw		clear all;
		A=imread('peppers.png','png');
		subplot(1,3,1);imshow(A);
		Agray=rgb2gray(A);
		subplot(1,3,2);imshow(Agray);
		Abw=im2bw(Agray,0.5);
		subplot(1,3,3);imshow(Abw);
			

imresize		clear all;
		A=imread('images/peppers.png','png');
		subplot(1,2,1);imshow(A);
		C=imresize (A,0.5);
		subplot(1,2,2);imshow(C);
			

imcrop		clear all;
		A=imread('peppers.png','png');
		subplot(1,2,1);imshow(A);
		Acrop=imcrop(A,[140,140,100,100]);
		subplot(1,2,2);imshow(Acrop);
		imwrite(Acrop,'peppersc.png');
			

Bwselect
Opció seleccionar els elements que es toquen per les cantonades amb l'element seleccionat
		clear all;
		A=imread('image14.jpg','jpg');
		Agray=rgb2gray(A);
		Abin=Agray>128;
		Abininv=not(Abin);
		subplot(1,4,1);imshow(Abininv);
		% Seleccionar els que es toquen en les cantonades
		Asel=bwselect(8);
		subplot(1,4,2);imshow(Asel);
			

3.2. Assajos

3.2.1. Assajos amb imatges extretes de la base de test

Per realitzar els assajos es disposava d'una base de dades amb vectors corresponents a 500 imatges de cadascun dels dígits del 0 al 9 escrits a mà. Aquesta base ha sigut utilitzada durant tot el treball per comparar imatges i determinar de forma estadística a quin dígit corresponien. Les imatges patró es troben en l'Annex 3 – Imatges patró.

Sobre una base de prova de 200 imatges de cada dígit es va fer un test que va donar com a resultat que el 89,75% de les imatges eren correctament classificades per l'script. Aquelles imatges que no eren detectades corresponien a dígits difícils d'identificar inclús per un ser humà. Aquestes imatges de prova es troben en l'Annex 4 – Imatges de prova.

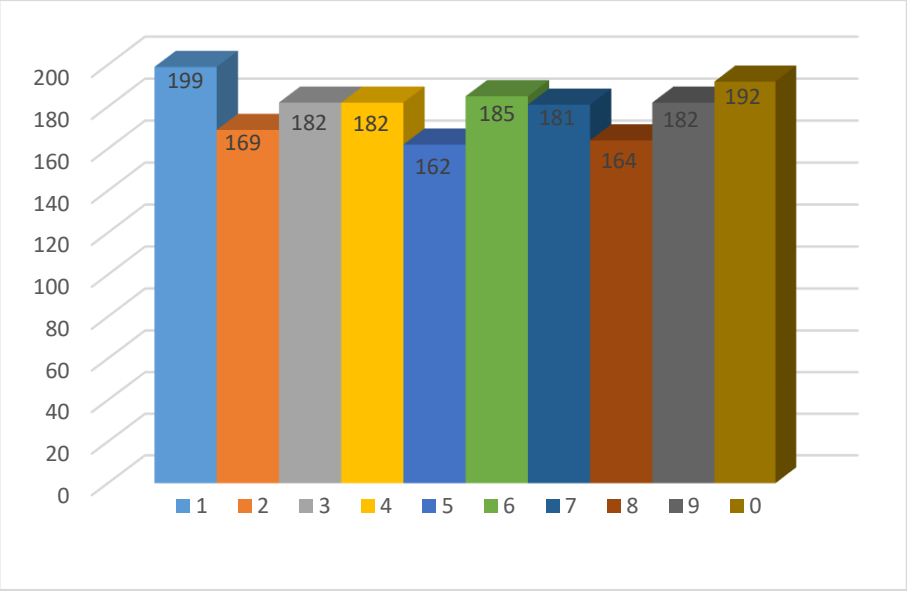
En l'Annex 5 – Script verificació 200 imatges, es troba el codi del script de *Matlab* utilitzat per verificar les 200 mostres de cada dígit i que va donar com a resultats els mostrats en la taula 3 de correspondències on cada fila correspon a les 200 mostres analitzades per cada dígit, i les columnes al número assignat per l'script classificador.

Taula 3. Resultat del reconeixement de 200 imatges de prova de cada dígit

	Correspondències										
Dígit	1	2	3	4	5	6	7	8	9	0	% Correctes
1	199	0	0	0	1	0	0	0	0	0	**99,5**
2	0	169	8	8	1	2	4	8	0	0	84,5
3	0	0	182	1	5	0	2	8	1	1	91
4	2	2	0	182	0	1	0	3	10	0	91
5	0	0	21	4	162	1	0	4	4	4	81
6	1	2	0	1	5	185	0	3	0	3	92,5
7	2	0	1	5	1	0	181	0	9	1	90,5
8	0	1	16	6	6	0	1	164	3	3	82
9	1	0	0	8	0	0	7	2	182	0	91
0	0	0	3	0	0	4	0	1	0	192	96

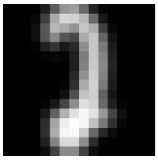
S'observa que el número 1, per raons derivades de la seva simplicitat morfològica, és el que millor resposta obté amb un percentatge del 99,5%. El segueixen el 0 i el 6. El més complicat de classificar és el 5.

Per altra banda observem també que el número 3 és el que més resultats obté com a candidat en els casos d'error ja que fins a 49 casos son classificats erròniament com a 3. En la imatge 14 podem veure una gràfica del número d'encerts per dígit extreta de la taula 3.



Imatge 14. Número d'encerts de cada dígit

En la imatge 15 es pot observar un dígit de la base de test que no va ser reconegut per l'script.



Imatge 15. Un dígit 2 no reconegut

En l'Annex 6 – Imatges no classificades es poden observar la morfologia dels dígits no classificats.

En la taula 4 es mostren, per cada dígit, els números que més vegades s'assignen per error al classificar un dígit. Es pot observar com el 3 i el 8 es confonen freqüentment entre ells. Passa el mateix amb el 4 i el 9. Aquestes confusions són a causa de les semblances morfològiques.

Taula 4. Número amb que es confon amb més freqüència cada dígit

Dígit	Màxima confusió
1	5
2	3, 4 i 8
3	8
4	9
5	3
6	5
7	9
8	3
9	4
0	6

Donat que aquestes mostres partien d'un format adequat a les dimensions de la base de dades, el següent repte va ser realitzar la comprovació amb imatges obtingudes amb un dispositiu extern (mòbil o similar).

3.2.2. Assajos amb imatges externes

La segona tanda de proves es va fer amb imatges obtingudes amb un dispositiu mòbil o mitjançant l'eina d'adquisició d'imatges de *Matlab, Image Acquisition Tool*.

En aquest assaig el problema que es va plantejar era el contrast i la discordança entre les mides de la imatge capturada i la base de dades de patrons utilitzada per fer l'anàlisi discriminatori. Per tant, per poder realitzar amb èxit aquest punt es van necessitar aplicar moltes de les funcions de *Matlab* enumerades anteriorment.

El tractament de la imatge va requerir de les següents passes prèvies a la seva comprovació:

1. Bolcar la imatge en una matriu (*imread*).
2. Adequar la dimensió de la imatge a un format quadrat de 1024x1024 (*imresize*).
3. Convertir-la a escala de grisos (*rgb2gray*).
4. Passar a binari la imatge amb un llindar de 128 sobre 255 (valor mig de l'escala de grisos).
5. Invertir la imatge per deixar el fons negre i el traç blanc.
6. Dilatar la imatge (*imdilate*) per reforçar la resolució del traç.
7. Enquadrar la imatge i centrar-la.
8. Ajustar la imatge a les dimensiones utilitzades per buscar en els patrons (16x16).
9. Realitzar una nova dilatació, ja que el canvi de mida reduïa el grossor del traç.

Es pot observar que no es fa una conversió directa a les dimensions de les imatges patró (16x16) ja que així s'evita la pèrdua de píxels amb informació clau per la classificació. La transició prèvia per una mida superior (1024x1024), amb més definició, permet ajustar condicions de la imatge i reforçar alguns aspectes com el traç abans de fer la reducció final a la dimensió 16x16 que tenen les imatges de la base de dades de patrons.

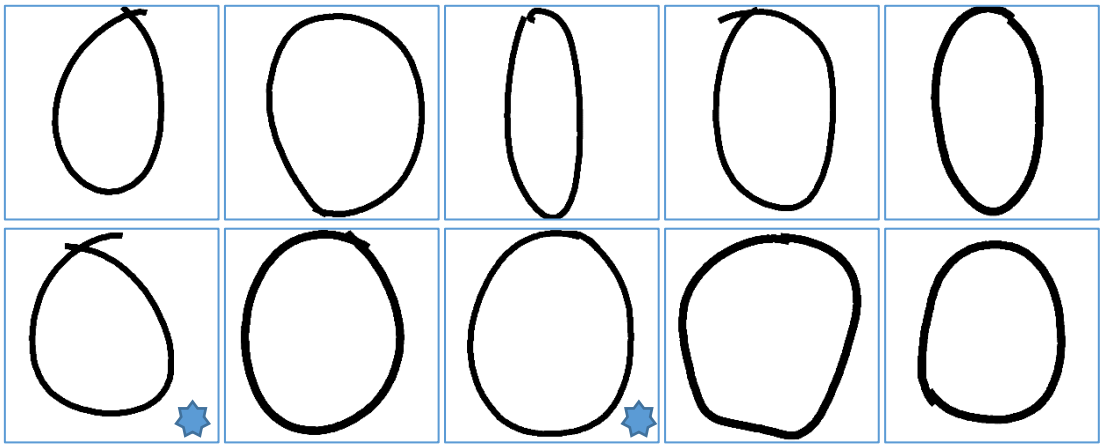
Es va observar en aquest procés que aconseguir un traç gruixut era clau per a la identificació, cosa que va conduir a l'ús de sentències repetitives de dilatació del tipus *imdilate*. Els resultats utilitzant línies de poc gruix i enquadrat no ajustat era prop d'un 50% inferior respecte als casos en què s'utilitzaven imatges ben enquadrades i amb el traç ample. Per aquest motiu es va incorporar una subrutina d'enquadrament de les imatges.

L'script realitzat per a la verificació d'imatges corresponents a diferents versions dels dígits del 0 al 9 es troba en l'Annex 7 – Script d'identificació d'imatges i s'explica pas a pas en l'apartat "2.2.3. Explicació pas a pas de l'script utilitzat pel reconeixement de dígits".

Cal destacar que com estem utilitzant un mètode estadístic, en l'script no sols s'obté el número que es correspon en més casos patró amb la imatge, sinó també el segon número amb més coincidències. La utilitat de conèixer el segon dígit amb més correspondències ens serveix per corroborar que en els casos d'error el número correcte es troba la majoria de les vegades en la segona posició.

A continuació podem veure una sèrie de dígits externs i quin va ser el resultat de l'anàlisi.

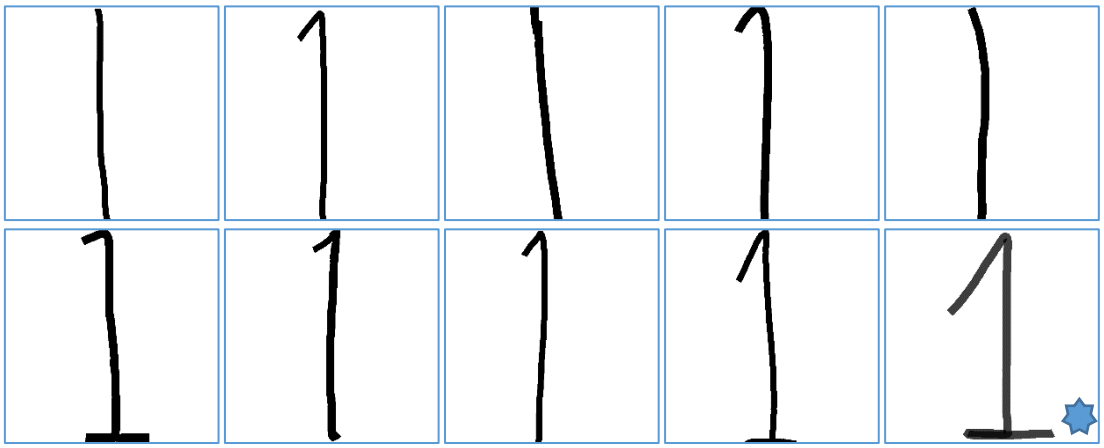
3.2.2.1. Comprovacions amb el dígit 0.



Imatge 16. Les 10 imatges externes del dígit 0

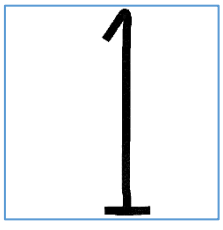
Pels elements mostrats a la imatge 16, corresponents a 10 versions del dígit 0, va haver un percentatge d'encert del 80%, sent les versions 5 i 7 (marcades amb una estrella) els casos erronis. En aquest casos erronis l'script va assimilar els dígits amb el 8 i el 4.

3.2.2.2. Comprovacions amb el dígit 1.



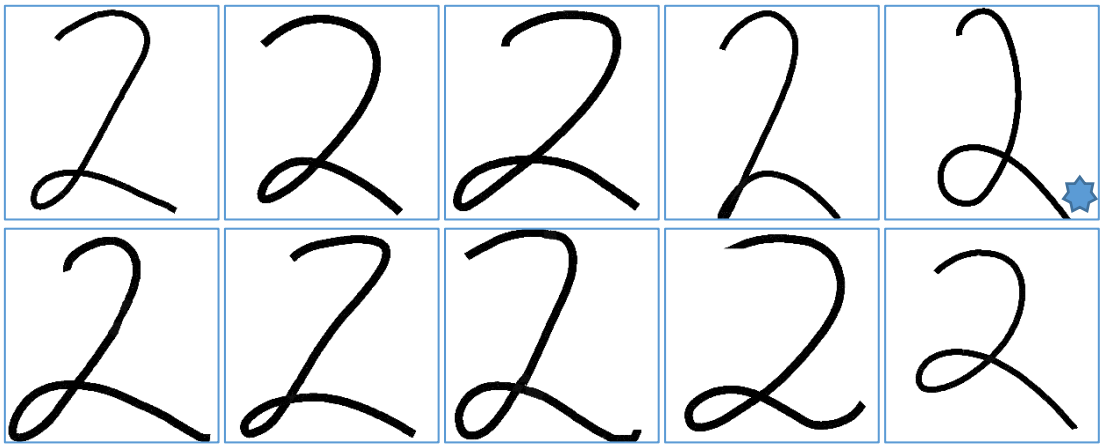
Imatge 17. Les 10 imatges externes del dígit 1

Pels elements mostrats a la imatge 17, corresponents a 10 versions del dígit 1, va haver un percentatge d'encert del 90%, sent la versió 9 del dígit (marcat amb una estrella) el cas erroni. Cal assenyalar que el dígit erroni va passar a ser correcte amb una correcció de la mida del peu i del barret, confirmant així que pel mètode es vital que les mostres estiguin el més a prop possible de la mitja. La imatge del dígit 1 corregit és la següent:



Imatge 18. Versió 9 del dígit 1 correctament classificada

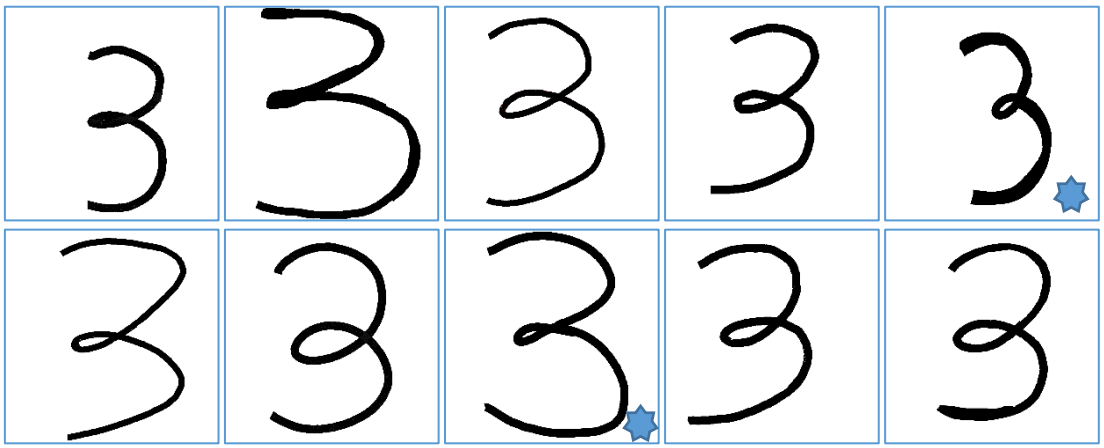
3.2.2.3. Comprovacions amb el dígit 2.



Imatge 19. Les 10 imatges externes del dígit 2

Pels elements mostrats a la imatge 19, corresponents a 10 versions del dígit 2 es va obtenir un encert del 90%. La versió incorrectament classificada va ser la quarta, marcada amb una estrella. Cal assenyalar que el dígit 2, junt amb el 5 i el 8 és el que donava més percentatge d'errors en les proves realitzades inicialment. En aquest grup milloraven els encerts si la base era amb el llaç. La versió errònia es confonia amb el 5.

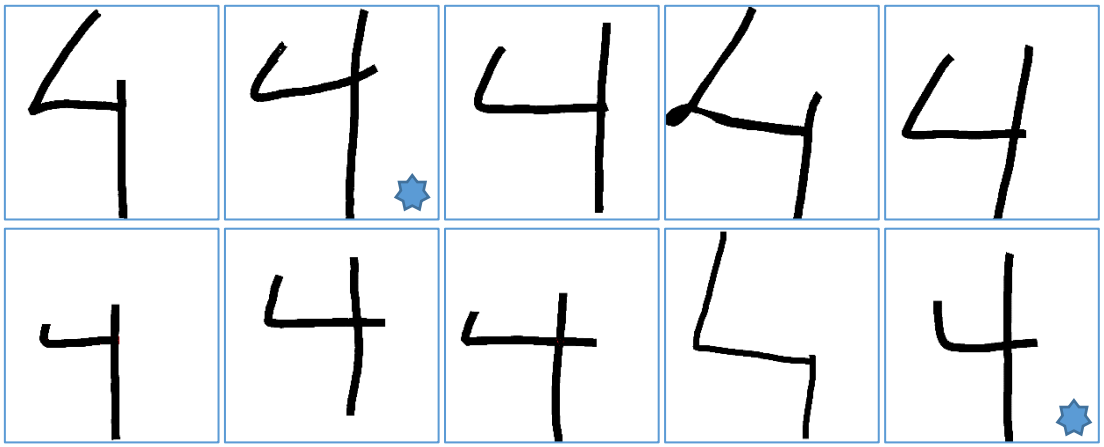
3.2.2.4. Comprovacions amb el dígit 3.



Imatge 20. Les 10 imatges externes del dígit 3

Pels elements mostrats a la imatge 20, corresponents a 10 versions del dígit 3 es van classificar correctament 8 versions i com a segona opció dos d'elles (marcades amb una estrella), situant-se per tant el 100% en les 2 primeres posicions. La confusió va ser amb el 0 i el 9.

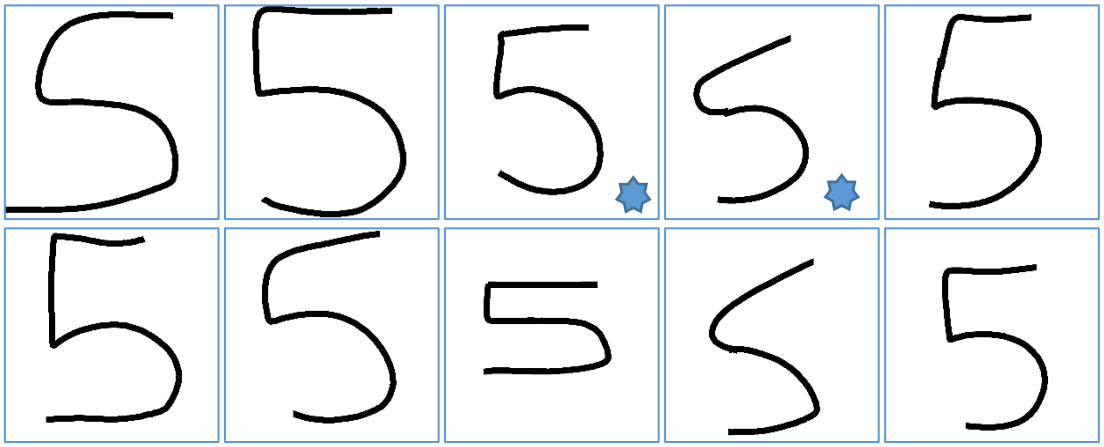
3.2.2.5. Comprovacions amb el dígit 4.



Imatge 21. Les 10 imatges externes del dígit 4

Pels elements mostrats a la imatge 21, corresponents a 10 versions del dígit 4, va haver un percentatge d'encert del 80%, sent les versions 1 i 9 del dígit (marcades amb una estrella) el casos erronis.

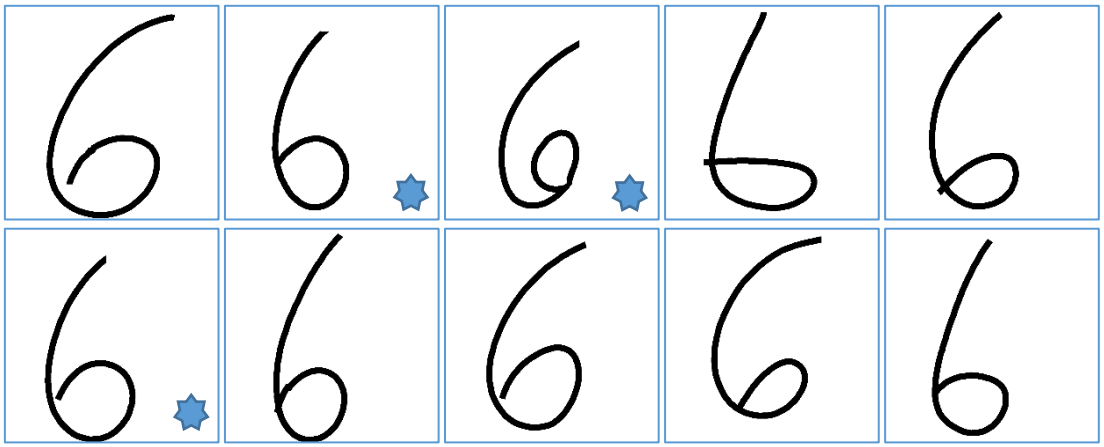
3.2.2.6. Comprovacions amb el dígit 5.



Imatge 22. Les 10 imatges externes del dígit 5

Pels elements mostrats a la imatge 22, corresponents a 10 versions del dígit 5 va haver un percentatge d'encert del 80%, sent les versions 2 i 3 del dígit (marcades amb una estrella) els casos erronis. En aquest dos casos erronis l'script va assimilar els dígits amb el 3 i el 9, i el 5 ocupava la segona i tercera posició.

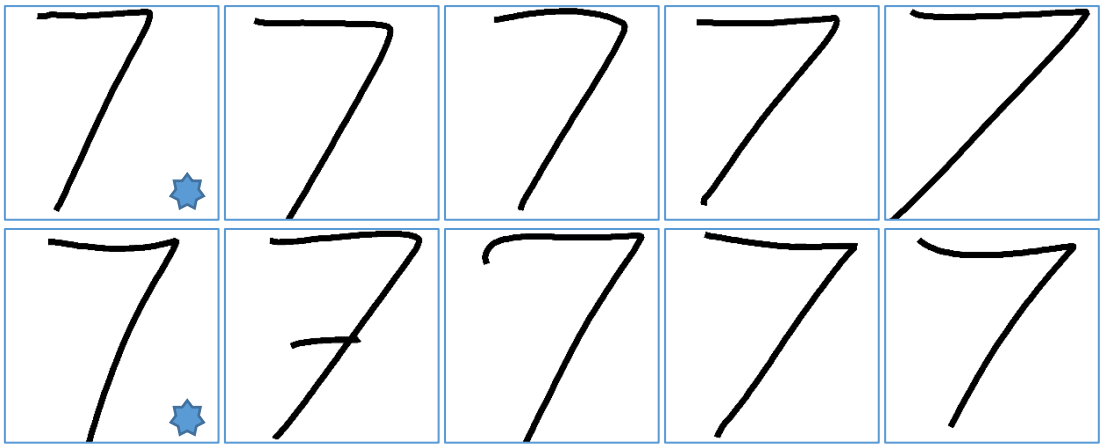
3.2.2.7. Comprovacions amb el dígit 6.



Imatge 23. Les 10 imatges externes del dígit 6

Pels elements mostrats a la imatge 23, corresponents a 10 versions del dígit 0, va haver un percentatge d'encert del 70%, sent les versions marcades amb una estrella els casos erronis. En aquest tres casos erronis l'script va assimilar els dígits amb els valors 2, 8 i 2, i en segona posició va classificar-los com a 6.

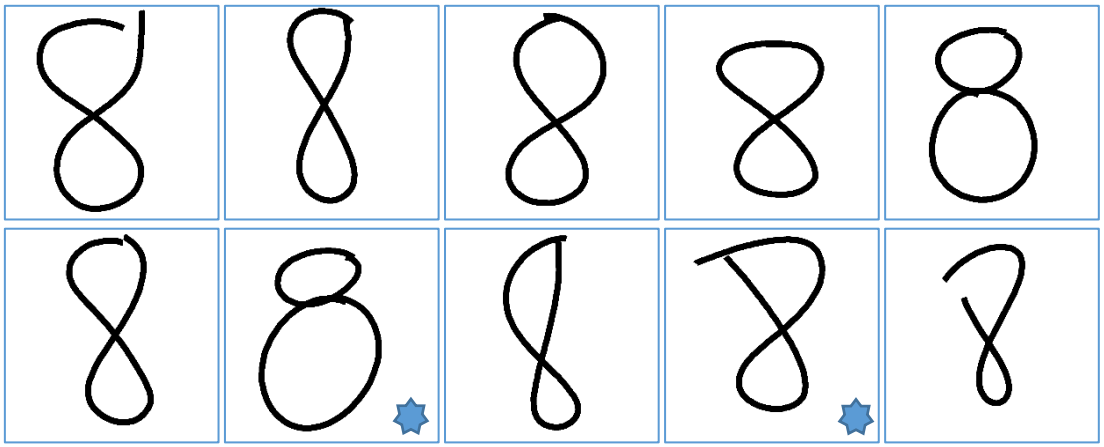
3.2.2.8. Comprovacions amb el dígit 7.



Imatge 24. Les 10 imatges externes del dígit 7

Pels elements mostrats a la imatge 24, corresponents a 10 versions del dígit 7, va haver un percentatge d'encert del 80%, sent les versions 0 i 5 del dígit (marcades amb una estrella) els casos erronis. En aquest casos erronis l'script va assimilar els dígits amb el 6 i el 7 ocupava la 2ª posició.

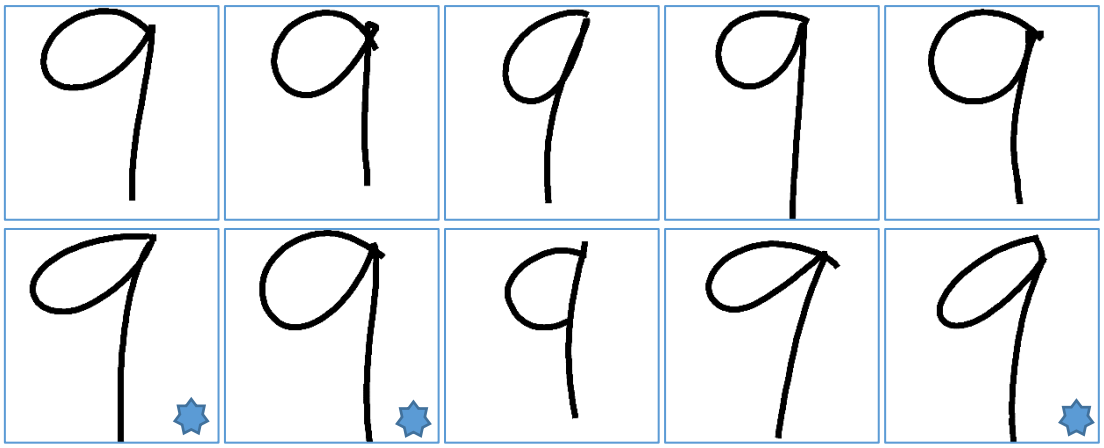
3.2.2.9. Comprovacions amb el dígit 8.



Imatge 25. Les 10 imatges externes del dígit 8

Pels elements mostrats a la imatge 25, corresponents a 10 versions del dígit 8, va haver un percentatge d'encert del 80%, sent les versions 6 i 8 del dígit (marcades amb una estrella) els casos erronis. En aquest casos erronis l'script va assimilar els dígits amb el 9, i el 8 ocupava posicions molt llunyanes.

3.2.2.10. Comprovacions amb el dígit 9.



Imatge 26. Les 10 imatges externes del dígit 9

Pels elements mostrats a la imatge 26, corresponents a 10 versions del dígit 9, va haver un percentatge d'encert del 70%, sent les versions 5, 6 i 9 del dígit (marcades amb una estrella) els casos erronis.

3.2.3. Explicació de l'script utilitzat pel reconeixement de dígits

% Rutina que tracta una imatge i fa el reconeixement del dígit.

```
clear all;                      % Inicialitzem totes les variables
load digit.mat X T;             % Carreguem en memòria la base de patrons i la de test
[d,m,c]=size(X);                % Guardem les mides de la base de patrons
S=zeros(d,d);                   % Netegem la matriu un guardarem els valor de covariància
for y=1:c                       % Recorrem la base de patrons
    mu(:,y)=mean(X(:,:,y),2);   % Calcular la mitjana per cada dígit (mi d'equació [3])
    S=S+cov(X(:,:,y)')/c;       % Calcular la matriu de covariància (S d'equació [3])
end
h=inv(S)*mu;                    % Matriu auxiliar per al càlcul de classe posterior
% Preguntar per la imatge a tractar
Numinp=input('Número a validar: ');
ver=input('Versió a validar: ');

% Preparar el nom de l'arxiu
filenam1=char(string(Numinp));filenam2=char(string(ver));filenam3=char(string('.png'));
filenam4=char(strcat(filenam1,filenam2,filenam3));
filenam5=char(string('images-'));
dirfile=char(strcat(filenam5,filenam1));
fullFileName = fullfile(dirfile,filenam4);

%Llegir l'arxiu i col·locar-lo en una matriu
A=imread(fullFileName);

% Mostrar la imatge llegida
imshow(A);
```

Imatge 27. La imatge del 3 que es vol validar

```
% Ajustar imatge a 1024x1024 i convertir-la en binària
A1=imresize(A,[1024,1024]);
Agray=rgb2gray(A1);
Abin=Agray>128;
Abininv=not(Abin);
imshow(Abininv);
```

Imatge 28. La imatge del 3 binària i invertida

```
% Dilatar la imatge per evitar pèrdua de píxels en l'ajust a 16x16
SE=strel('square',10);
Adilate=Abininv;
for dil=1:2
    Adilate=imdilate(Adilate,SE);
end
imshow(Adilate);
```

Imatge 29. La imatge del 3 dilatada

```
% Enquadrar la imatge
Adilate=Enquadra(Adilate);
imshow(Adilate);
```

Imatge 30. La imatge del 3 enquadrada

```
% Ajustar la imatge a 16x16
Adilate=imresize(Adilate,[16,16]);
TE=strel('square',2);
```

Imatge 31. La imatge del 3 passada a 16x16

```
% Tornar a dilatar la imatge per ressaltar píxels després de l'ajust a 16x16
Adilate=imdilate(Adilate,TE);
imshow(Adilate);
```

Imatge 32. La imatge del 3 tornada a dilatar

```
% Convertir imatge a format patrons en què els valor oscil·len en el rang [-1,1]
Adilate=Adilate.*255;
Areal=double(Adilate);
Areal2=Areal.*2;
Areal3=Areal2./255;
Areal4=Areal3-1;
Areal5=Areal4.';

% Passar imatge a Vector
E=Areal5(:);   % x^T de l'equació [3]

% Creació de la taula amb el valor del terme de classe posterior (Equació [3])
p(:,:,1)=h'*E(:,:,1)-repmat(sum(mu.*h)',[1,m])/2;

% Matriu "q" en què guardarem els resultats del terme de decisió de la classe a posteriori
% per després ordenar-los de major a menor en la matriu "qsort"
q=zeros(c);
for y=1:c
    yleer=y;
    if y==10
        yleer=0;
    end
    q(y,1)=yleer;
    q(y,2)=p(y,1,1);
end
```

1	1.2517e+04
2	1.2611e+04
3	1.2678e+04
4	1.2631e+04
5	1.2580e+04
6	1.2541e+04
7	1.2670e+04
8	1.2648e+04
9	1.2625e+04
0	1.2675e+04

Valor amb terme de decisió de la classe a posteriori més alt

Imatge 33. La matriu "q" on es veu el valor del terme de decisió per cada dígit

```
% Ordenem els resultats per ordre de màxima terme de decisió de la classe a posteriori
qsort=sortrows(q,2,'descend');
```

3	1.2678e+04
0	1.2675e+04
7	1.2670e+04
8	1.2648e+04
4	1.2631e+04
9	1.2625e+04
2	1.2611e+04
5	1.2580e+04
6	1.2541e+04
1	1.2517e+04

Valor amb terme de decisió més alt

Imatge 34. La matriu "qsort" amb els valors del terme de decisió de la classe a posteriori ordenats de major a menor

```
% Mostrar resultats
Numero=qsort(1,1);
lit0=char('Imatge és: ');lit1=char(string(Numinp));lit1v=char(' v:');lit2=char(string(ver));
lit3=char(string(' 1ª: '));lit4=char(string(Numero));
lit5=char(string(' 2ª: '));lit6=char(string(qsort(2,1)));
lit7=char(string(' 3ª: '));lit8=char(string(qsort(3,1)));
lit9=char(string(' 4ª: '));lit10=char(string(qsort(4,1)));
lit11=char(string(' 5ª: '));lit12=char(string(qsort(5,1)));
lit13=char(string(' 6ª: '));lit14=char(string(qsort(6,1)));
lit15=char(string(' 7ª: '));lit16=char(string(qsort(7,1)));
lit17=char(string(' 8ª: '));lit18=char(string(qsort(8,1)));
lit19=char(string(' 9ª: '));lit20=char(string(qsort(9,1)));
lit21=char(string(' 10ª: '));lit22=char(string(qsort(10,1)));
litn=char('\n');
lit=char(strcat(lit0,lit1,lit1v,lit2,lit3,lit4,lit5,lit6,lit7,lit8,lit9,lit10,lit11,lit12,lit13,lit14,lit15,lit16,lit17,lit18,lit19,lit20,lit21,lit22,litn));
fprintf(lit);
```

```
1ª:3  2ª:0  3ª:7  4ª:8  5ª:4  6ª:9  7ª:2  8ª:5  9ª:6  10ª:1
```

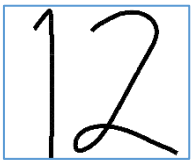
Imatge 35. Resultats ordenats pel valor del terme de decisió

3.2.4. Assaig amb imatges amb més d'un dígit

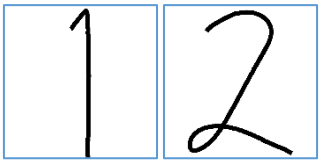
Per a complementar el treball es va fer un assaig amb imatges amb dos dígits no enllaçats entre ells. El procés simplement va consistir en dividir prèviament la imatge en tantes imatges individuals com dígits contingués. Un cop feta la separació el procés a utilitzar era el mateix que en el tractament d'imatges individuals.

Podem veure a continuació dos exemples.

En l'exemple primer separem 2 dígits.



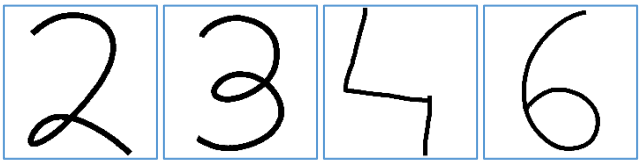
Imatge 36. El número 12 abans de fer la separació.



Imatge 37. Els 2 dígits del número 12 després de fer la separació.

En l'exemple segon separem 4 dígits.

Imatge 38. El número 2346 abans de fer la separació.



Imatge 39. Els 4 dígits del número 2346 després de fer la separació.

L'script utilitzat per separar els dígits es troba en l'Annex 8 – Script de separació de dígits.

A continuació es detallen les sentències de l'script de separació.

```
% Rutina per tractar una imatge i separar-ne els dígits.
Clear all;
% Preguntar per la imatge a separar.
Numinp=input('Número a separar: ');
ver=input('Versió a separar: ');
% Preparar el nom de l'arxiu i llegir-lo
filenam1=char(string(Numinp));filenam2=char(string(ver));filenam3=char(string('.png'));
filenam4=char(strcat(filenam1,filenam2,filenam3));
filenam5=char(string('images-'));
dirfile=char(strcat(filenam5,filenam1));
fullFileName = fullfile(dirfile,filenam4);
A=imread(fullFileName);
% Passar a escala de grisos i binaritzar la imatge;
Agray=rgb2gray(A);
Abin=Agray>128;
[altoimagen,amplada]=size(Abin);        %Capturem les mides de la imatge original
numdigits=0;                            % Comptador amb numero de imatges trobades
colini=zeros(10);                       % Guardarem aquí la posició inicial de cada dígit trobat
colfin=zeros(10);                       % Guardarem aquí la posició final de cada dígit trobat
imatgetrobada=0;
% Recorregut per analitzar totes les columnes
for amp=1:amplada
    if Abin(:,amp)==1                   % Si trobem una fila buida
        if imatgetrobada==1             % i estem recorrent un dígit,
          imatgetrobada=0;              % netegem indicador de dígit trobat
          colfin(numdigits)=amp-1;      % i guardem la posició anterior com a final d'un dígit
        end
    else                                % Si trobem una fila no buida
        if imatgetrobada==0             % Si a la posició anterior no havia imatge
          imatgetrobada=1;              % Activem indicador de dígit trobat
          numdigits=numdigits+1;        % incrementem comptador de dígits trobats
          colini(numdigits)=amp;        % i guardem la posició com a inici d'un dígit
        end
    end
end
% Recorregut per retallar els dígits i gravar-los en arxius separats
for dig=1:numdigits
    final=colfin(dig)-colini(dig)+1;        % Calculem amplada dígit
    digito=imcrop(Abin,[colini(dig),1,final,altoimagen]); % Tallem dígit
    [altodigito,ampladadigito]=size(digito);
    % Traslladar el dígit a matriu quadrada si la forma no ho és
    if altodigito>ampladadigito
      digito2=ones(altodigito,altodigito);
      diferencia=round((altodigito-ampladadigito)/2);
      amplada2=ampladadigito+diferencia-1;
      indice=1;
      for despl=diferencia:amplada2
          digito2(:,despl)=digito(:,indice);
```

```
            indice=indice+1;
        end
    else
        digito2=digito;
    end
    % Conversió de imatge binària a RGB
    grayImage = 255 * uint8(digito2);
    RGB = cat(3, grayImage, grayImage, grayImage);
    % Codi per preparar el nombre de l'arxiu i
    % gravar-lo per a posterior processament amb rutina de classificació
    filenam1=char(string(Numinp));filenam2=char(string(ver));filenam3=char(string('.png'));
    filenam4=char(strcat(filenam1,filenam2,filenam3));
    filenam5=char(string('images-'));
    dirfile=char(strcat(filenam5,filenam1,filenam1));
    mkdir(dirfile);
    fullFileName = fullfile(dirfile,filenam4);
    imwrite(RGB,fullFileName);
end
```

4. ANÀLISI DE L'IMPACTE AMBIENTAL

La utilització de tècniques de reconeixement produeix un estalvi en el consum de paper quan l'apliquem a situacions en què tradicionalment s'emprava el paper com a eina d'identificació. La identificació directa amb us de dispositius mòbils o càmeres evita el registre a formularis en paper.

No es considera cap impacte ambiental derivat del mal funcionament de les rutines de reconeixement, considerant que si bé els errors en la classificació poden provocar temps d'espera i tenir conseqüències econòmiques negatives, no es detecten possibles impactes sobre l'entorn natural.

5. PRESSUPOST I/O ANÀLISI ECONÒMICA

El pressupost del projecte està desglossat en tres blocs:

1. Serveis professionals: Contempla el cost de totes les activitats realitzades en les fases d'anàlisi i implementació del projecte. Aquestes es divideixen en tres categories per tal d'assignar el recurs idoni i optimitzar el cost del projecte:
    - Consultoria i anàlisi, realitzades per personal amb alta qualificació funcional. Per aquesta activitat s'assigna un perfil Consultor funcional.
    - Implementació, realitzades per personal amb qualificació tècnica. Serà realitzar per un Programador o Desenvolupador
    - Proves: que poden ser realitzades per personal amb qualificació mitja. S'assigna un perfil "Tester" per aquesta activitat.
2. Programari: Contempla el cost dels elements de programari necessaris per a desenvolupar les següents tasques:
    - Part teòrica: recerca, documentació, espai compartit: Suite office 365.
    - Part pràctica: Programari Matlab
    - Proves: Programari Matlab
3. Maquinari: Contempla el cost dels elements de maquinari necessaris per a desenvolupar les tasques del projecte.

En la taula 5 es mostra el pressupost desglossat en els tres blocs principals detallant l'esforç per perfil professional i per cadascuna de les parts de maquinari i programari necessàries.

Taula 5. Pressupost detallat del projecte en Euros

Pressupost del projecte en Euros					
Serveis professionals					
Perfil	Activitat	Hores	Cost/hora	Cost	Comentaris
Consultor funcional	Consultoria i anàlisi funcional	228	35	7.980,00	Tarifa d'empresa del sector TIC
Desenvolupador	Implementació d'scripts Matlab	266	25	6.650,00	Tarifa d'empresa del sector TIC
Tester	Proves amb imatges	124	20	2.480,00	Tarifa d'empresa del sector TIC
			Total Serveis professionals	**17.110,00**	
Programari					
Programa	Activitat	Unitats	Preu Llicència	Cost	Comentaris
Office 365	Documentació i espai comú	3	12,9	38,70	Preu llicència professional mensual
Matlab	Desenvolupament i proves	2	8	16,00	Preu llicència professional anual
Image Acquisition Tool	Adquisició d'imatges	1	4	4,00	Preu llicència professional anual
			Total Programari	**58,70**	
Maquinari					
Dispositiu	Activitat	Unitats	Preu*	Cost	Comentaris
Ordinador portàtil	Documentació, anàlisi i proves	1	125	125,00	Surface amb programari bàsic
Telèfon mòbil	Adquisició d'imatges	1	58	58,00	Model Samsung S8
			Total Maquinari	**2.198,00**	
			Total Projecte	**17.351,70**	

* Cost del maquinari per un període de 4 mesos considerant una amortització de 4 anys

6. Conclusions

La primera i principal conclusió és que en el treball s'han assolit els objectius que s'havien definit a l'inici del mateix. L'abast del treball incloïa la identificació de dígits numèrics de forma individualitzada i aquest objectiu s'ha assolit amb resultats positius que oscil·laven entre el 80 i el 90%.

La segona conclusió del treball és que per aplicar aquesta mena de mètodes de reconeixement cal adequar les imatges a les mides i condicions dels patrons amb els quals les volem comparar. És a dir, que no podem simplement contrastar una imatge amb una base de patrons sinó que cal prèviament fer una sèrie d'accions entre les que destaquem les següents:

- Ajustar la mida externa (número de files i columnes)
- Preservar els paràmetres de la imatge que siguin claus per al procés d'identificació que volem dur a terme (En reconeixement de text escrit seria el contorn i gruix de la línia).
- Convertir la imatge a format binari per eliminar el soroll de la imatge.

Per tant, és molt important comptar amb eines potents de gestió d'imatges per poder fer tots els passos previs de preparació de forma prou ràpida perquè puguem processar i identificar imatges en temps quasi real.

La tercera conclusió és que aquests mètodes poden ser duts a la indústria per detectar oportunitats d'aplicació pràctica que obrin les portes al desenvolupament de projectes complexos. Els camps d'aplicació de mètodes avançats de reconeixement d'informació poden ser útils per ampliar prestacions en el camps de la seguretat (tant de la informació com la personal) o per aplicar-los en el terreny de la investigació científica per contrastar patrons d'informació mèdica o biomèdica.

La quarta conclusió del treball és que *Matlab* és una eina molt potent per a la gestió de grans quantitats d'informació. Una dada significativa és que l'script processat de les 200 imatges de prova dura 6.94 segons, que dona una mitjana de 0,0347 segons per imatge. Si comptem que per cada imatge s'utilitzen 60 sentències principals, tenim un temps de $5{,}78 \times 10^{-4}$ segons per sentència. Una altra dada interessant és que una rutina per gravar en una carpeta del sistema 500 imatges en format *png*[2], després de processar cadascuna d'elles en blocs de 25 sentències, triga tan sols 18 segons.

La darrera conclusió és que com a continuïtat d'aquest projecte seria interessant estudiar l'aplicació de rutines més avançades com "*Cross Validation*", tècnica utilitzada per avaluar els resultats d'una anàlisi estadística que vol garantir que els resultats són independents de la partició entre dades d'entrenament i prova, i que dona resultats un 6% millors que el mètode FDA.

[2] Acrònim de *Portable Network Graphics*, format gràfic basat en un algoritme de compressió sense pèrdua per *bitmaps* no subjecte a patents (Font: Viquipèdia)

7. Bibliografia

[1] Prácticas Complementarias al curso de Matemáticas II al tema de las derivadas parciales. Leonardo Acho Zuppa. Abril 2015.

[2] https://www.ara.cat/opinio/Messi-gol-foto-Bernabeu-historia_0_1783621701.html

[3] An Introduction to Statistical Machine Learning. Masashi Sugiyama. Ed. Morgan. Kaufmann.2016

[4] Procesamiento de imágenes con *Matlab*. Disponible en: https://es.slideshare.net/lonely113/procesamiento-digital-de-imgenes-con-matlab

[5] Optical character recognition. https://en.wikipedia.org/wiki/Optical_character_recognition

[6] Análisis de datos multivariantes. Daniel Peña. 2002

[7] http://bigpictures.club/resize.php?img=http://image.slidesharecdn.com/main-120718022005-phpapp01/95/introduction-to-common-spatial-pattern-filters-for-eeg-motor-imagery-classification-10-728.jpg

[8] https://techstore.msu.edu/software/mathworks-matlab-student-license-0

8. Annexos

Annex 1 – Script Laplacià

Annex 2 – Script Diferenciador

Annex 3 – Imatges patró dels dígits

Annex 4 – Imatges de prova dels dígits

Annex 5 – Script verificació 200 imatges

Annex 6 – Imatges no classificades

Annex 7 – Script d' identificació d'imatges

Annex 8 – Script de separació de dígits

Annex 1 – Script Laplacià

```
% Inicialitzar variables
clear all;
% Llegir imatge
A1=imread('image3.jpg','jpg');
% Aplicar escala de grisos
A=0.21*A1(:,:,1)+0.72*A1(:,:,2)+0.07*A1(:,:,3);
% Adquirir mida imatge
[r,c,k]=size(A);
% Aplicar rutina a cada píxel. Es calcula un nou valor sumant-li els
% valors dels píxels de les cantonades i restant-li 4 del seu propi valor
for j=2:c-1
      for i=2:r-1
          B(i,j)=A(i-1,j)+A(i,j-1)-4*A(i,j)+A(i,j+1)+A(i+1,j);
      end
end
% Mostrar imatge original i modificada
subplot(1,2,1);imshow(A);
subplot(1,2,2);imshow(B);
% Gravar imatge en arxiu
imwrite(B,'image3a.jpg');
```

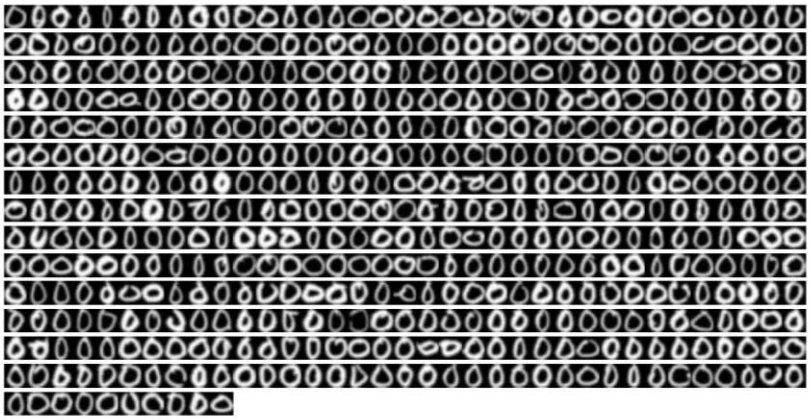
Annex 2 – Script Diferenciador

```
% Inicialitzar variables
clear all;
% Establir llindar de comparació
umbral=25;
% Llegir imatge
A1=imread('image2.jpg','jpg');
% Passem escala de grisos
A=0.21*A1(:,:,1)+0.72*A1(:,:,2)+0.07*A1(:,:,3);
% Adquirir mida imatge
[r,c,k]=size(A);
% Rutina de recorregut en diagonal en ambdós sentits
for j=2:c-1
      for i=2:r-1
          B1(i,j)=A(i,j)-A(i-1,j-1);
          B2(i,j)=A(i-1,j)-A(i,j+1);
      end
end
Bx=B1+B2;
% Comparador de llindar
for j=2:c-1
      for i=2:r-1
          if Bx(i,j)>umbral;
                    B(i,j)=0;
          else
                    B(i,j)=255;
          end
      end
end
% Mostrar imatge original i modificada
subplot(2,4,1);imshow(A);
subplot(2,4,2);imshow(B);
% Gravar imatge en arxiu
imwrite (B,'image2b.jpg');
```

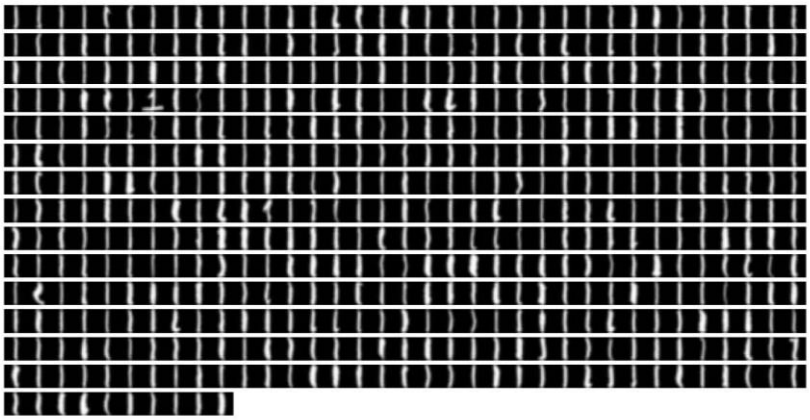
Annex 3 – Imatges patró dels dígits

A continuació es mostren els 500 dígits utilitzats com a patrons.

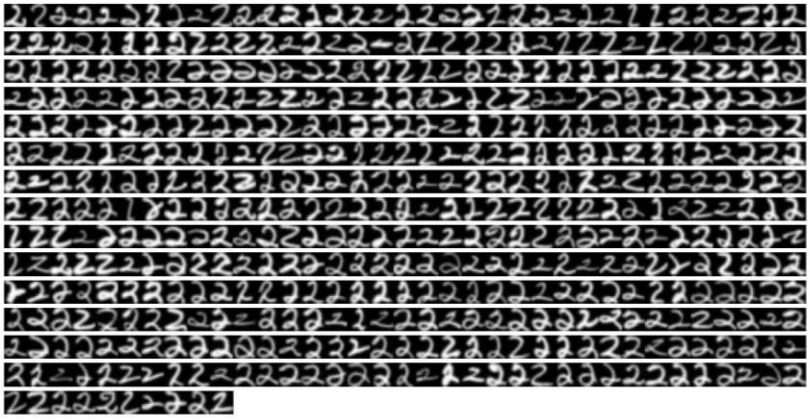
Dígits 0



Dígits 1

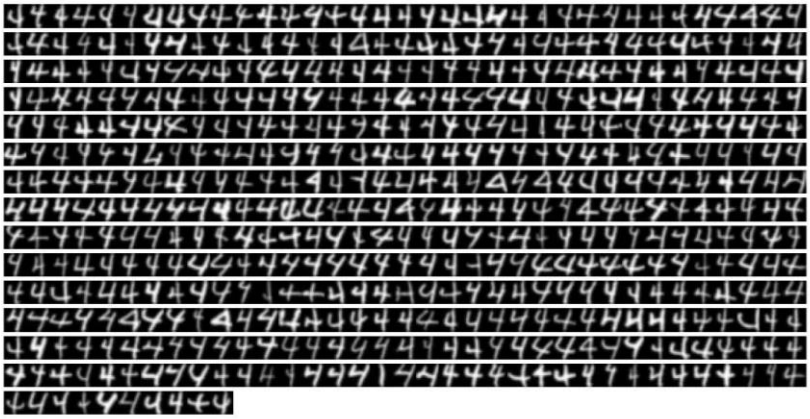


Dígits 2



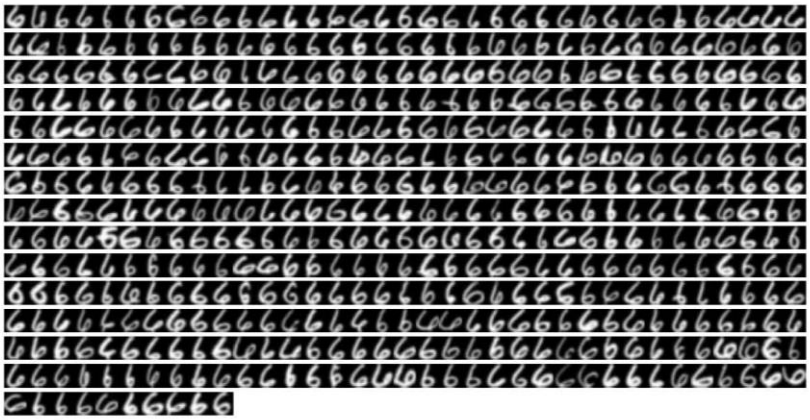
Dígits 3

Dígits 4

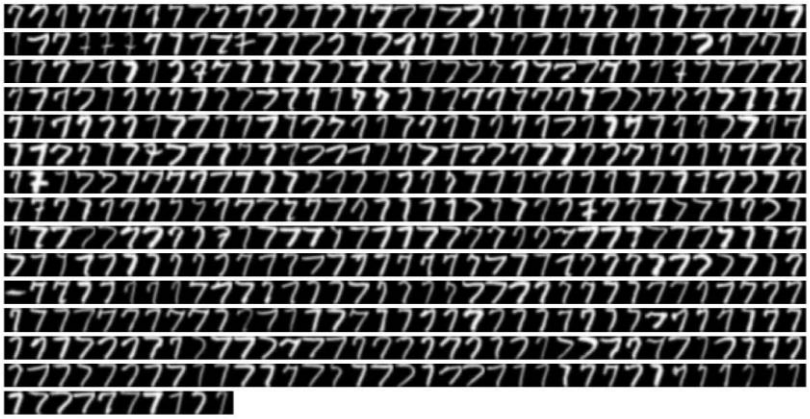


Dígits 5

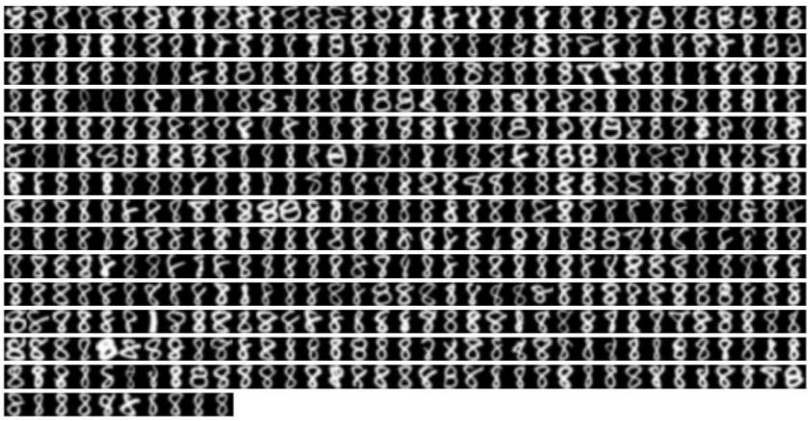
Dígits 6



Dígits 7



Dígits 8

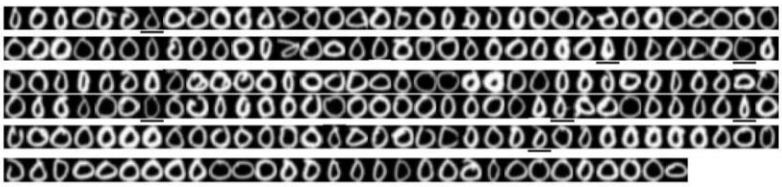


Dígits 9

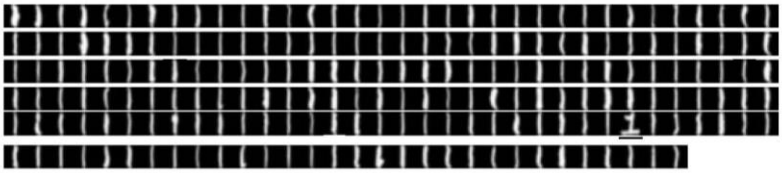
Annex 4 – Imatges de prova dels dígits

A continuació es mostren els 200 dígits utilitzats com a prova.
Es subratllen aquells que no han estat classificats correctament per l'script.

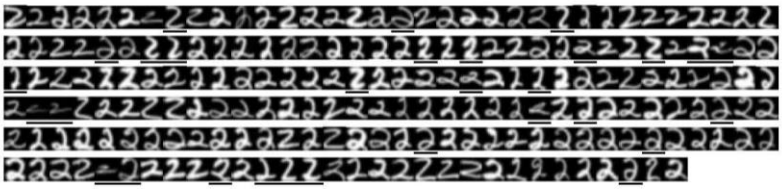
Dígits 0



Dígits 1



Dígits 2

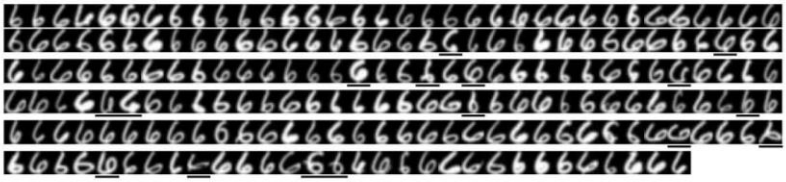


Dígits 3

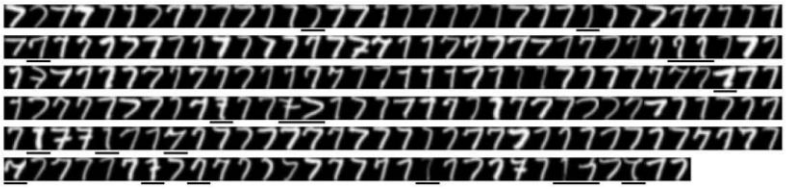
Dígits 4

Dígits 5

Dígits 6



Dígits 7



Dígits 8

Dígits 9

Annex 5 – Script verificació 200 imatges

```
% Rutina que valida totes les imatges extretes de la base d'imatges "Digit.mat"
% contra els 500 patrons que hi ha per a cada dígit.
Clear all;
% Carregar en memòria els patrons X i les imatges de test T
load digit.mat X T;
[d,m,c]=size(T);
S=zeros(d,d);
ERR=zeros(c,m);
correctos=0;
totales=0;
correctosn=0;
% Preparar taula de patrons per a la comparació
for y=1:c
      mu(:,y)=mean(X(:,:,y),2);
      S=S+cov(X(:,:,y)')/c;
end
% Bucle per validar els 10 dígits
for Numleer=0:9
      correctosn=0;
      inc=1;
      % Bucle per validar les 200 versions de cada dígit
      for v=1:200
          totales=totales+1;
          % Preparar el nom de l'arxiu i llegir-lo
          vstr=string(v);
          filenam1=char(string(Numleer));
          filenam2=pad(vstr,3,'left','0');
          filenam3=char(string('.png'));
          filenam4=char(strcat(filenam1,filenam2,filenam3));
          dirfile=char(string('Timages1-png'));
          fullFileName = fullfile(dirfile,filenam4);
          A=imread(fullFileName);
          % Convertir matriu d'imatge al format dels patrons per poder fer la comparació
          Areal=double(A);
          Areal2=Areal.*2;
          Areal3=Areal2./255;
          Areal4=Areal3-1;
          Areal5=Areal4.';
          E=Areal5(:);
          h=inv(S)*mu;


          % Comparació
          p(:,:,1)=h'*E(:,:,1)-repmat(sum(mu.*h)',[1,m])/2;
```

```
        % Matriu “q” en què guardarem els resultats per a després ordenar-los de major
        % a menor en la matriu “qsort”
        q=zeros(c);
        for y=1:c
                yleer=y;
                if y==10
                        yleer=0;
                end
        q(y,1)=yleer;
        q(y,2)=p(y,1,1);
        end
        qsort=sortrows(q,2,'descend');
        Numero=qsort(1,1);
        nleer=n;
        if n==10
                nleer=0;
        end
        if Numero == nleer
                correctos=correctos+1;
                correctosn=correctosn+1;
        else
                ERR(n,inc)=v;
                inc=inc+1;
        end
    end
    % Mostrar resultats
    formatSpec = 'Serie %1.0f. Correctos: %3.0f \n';
    fprintf(formatSpec,nleer,correctosn);
end
% Mostrar percentatge d’encerts
porcentaje=correctos*100/totales;
formatSpec = 'Global. Correctos: %3.0f. Porcentaje: %2.2f \n';
fprintf(formatSpec,correctos,porcentaje);
```

Annex 6 – Imatges no classificades

Dígits 0

Dígits 1

Dígits 2

Dígits 3

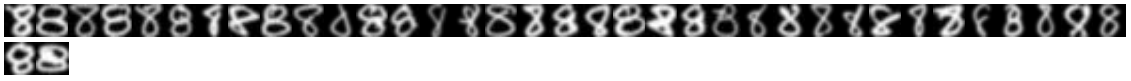
Dígits 4

Dígits 5

Dígits 6

Dígits 7

Dígits 8



Dígits 9

Annex 7 – Script d' identificació d'imatges

```
% Rutina que tracta una imatge i fa el reconeixement del dígit.
clear all;
load digit.mat X T;
[d,m,c]=size(T);
S=zeros(d,d);
for y=1:c
      mu(:,y)=mean(X(:,:,y),2);
      S=S+cov(X(:,:,y)')/c;
end
% Preguntar imatge a tractar
Numinp=input('Número a validar: ');
ver=input('Versió a validar: ');
% Preparar el nom de l'arxiu
filenam1=char(string(Numinp));filenam2=char(string(ver));filenam3=char(string('.png'));
filenam4=char(strcat(filenam1,filenam2,filenam3));
filenam5=char(string('images-'));
dirfile=char(strcat(filenam5,filenam1));
fullFileName = fullfile(dirfile,filenam4);
A=imread(fullFileName);
subplot(1,8,1); imshow(A);
% Ajustar imatge a 1024x1024 i convertir-la en binària
A1=imresize(A,[1024,1024]);
subplot(1,8,2); imshow(A1);
Agray=rgb2gray(A1);
subplot(1,8,3); imshow(Agray);
Abin=Agray>128;
subplot(1,8,4); imshow(Abin);
Abininv=not(Abin);
subplot(1,8,5);imshow(Abininv);
% Dilatar la imatge per evitar pèrdua de píxels en l'ajust a 16x16
SE=strel('square',10);
Adilate=Abininv;
for dil=1:2
      Adilate=imdilate(Adilate,SE);
end
subplot(1,8,6);imshow(Adilate);
% Enquadrar la imatge
Adilate=Enquadra(Adilate);
subplot(1,8,7);imshow(Adilate);figure(1);
% Ajustar imatge a 16x16
Adilate=imresize(Adilate,[16,16]);
TE=strel('square',2);
% Tornar a dilatar la imatge per ressaltar píxels després de l'ajust a 16x16
Adilate=imdilate(Adilate,TE);
subplot(1,8,8);imshow(Adilate);
```

```
% Convertir imatge a format patrons
Adilate=Adilate.*255;
Areal=double(Adilate);
Areal2=Areal.*2;
Areal3=Areal2./255;
Areal4=Areal3-1;
Areal5=Areal4.';
E=Areal5(:);
subplot(1,6,6);
imagesc(reshape(Areal4,[16,16]));
% Comparació
h=inv(S)*mu;
p(:,:,1)=h'*E(:,:,1)-repmat(sum(mu.*h)',[1,m])/2;
% Matriu "q" en què guardarem els resultats per després ordenar-los de major
% a menor en la matriu "qsort"
q=zeros(c);
for y=1:c
      yleer=y;
      if y==10
          yleer=0;
      end
      q(y,1)=yleer;
      q(y,2)=p(y,1,1);
end
% Ordenar els resultats per ordre del terme de decisió de la classe a posteriori
qsort=sortrows(q,2,'descend');
% Mostrar resultats
Numero=qsort(1,1);
lit0=char('Imagen es: ');lit1=char(string(Numinp));lit1v=char(' v:');lit2=char(string(ver));
lit3=char(string(' 1ª: '));lit4=char(string(Numero));
lit5=char(string(' 2ª: '));lit6=char(string(qsort(2,1)));
lit7=char(string(' 3ª: '));lit8=char(string(qsort(3,1)));
lit9=char(string(' 4ª: '));lit10=char(string(qsort(4,1)));
lit11=char(string(' 5ª: '));lit12=char(string(qsort(5,1)));
lit13=char(string(' 6ª: '));lit14=char(string(qsort(6,1)));
lit15=char(string(' 7ª: '));lit16=char(string(qsort(7,1)));
lit17=char(string(' 8ª: '));lit18=char(string(qsort(8,1)));
lit19=char(string(' 9ª: '));lit20=char(string(qsort(9,1)));
lit21=char(string(' 10ª: '));lit22=char(string(qsort(10,1)));
litn=char('\n');
lit=char(strcat(lit0,lit1,lit1v,lit2,lit3,lit4,lit5,lit6,lit7,lit8,lit9,lit10,lit11,lit12,lit13,lit14,lit15,lit16,li
t17,lit18,lit19,lit20,lit21,lit22,litn));
fprintf(lit);
```

Subrutina Enquadra

```
% 2018-04-15. Amb aquesta rutina s'ajusta la imatge verticalment
function imaenq=Enquadra(imasinenq)
Aconvert=imasinenq;
[altoimagen,amplada]=size(Aconvert);
% Ajustar imatge per la part superior
todosceroprimera=1; % Indicador per saber si la primera línia està buida o no.
% Fins que l'indicador no sigui 0 no para el bucle
while todosceroprimera==1
% Si la fila està buida retallem, si no, parem
    if Aconvert(1,:)==0
        altoimagen=altoimagen-1;
        Aconvert=imcrop(Aconvert,[1,2,amplada,altoimagen]);
    else
        todosceroprimera=0;
    end
end
[altoimagen2,amplada2]=size(Aconvert);
Aconvert2=Aconvert;
% Rutina per ajustar imatge per la part inferior
todosceroprimera=1;
while todosceroprimera==1
    if Aconvert2(altoimagen2,:)==0
        altoimagen2=altoimagen2-1;
        Aconvert2=imcrop(Aconvert2,[1,1,amplada2,altoimagen2]);
    else
        todosceroprimera=0;
    end
end
% Imatge que retorna la funció
imaenq=Aconvert2;
end
```

Annex 8 – Script de separació de dígits

```
% Rutina per tractar una imatge i separar-ne els dígits.
clear all;
Numinp=input('Número a validar: ');
ver=input('Versió a validar: ');
% Preparar el nom de l'arxiu i llegir-lo
filenam1=char(string(Numinp));filenam2=char(string(ver));filenam3=char(string('.png'));
filenam4=char(strcat(filenam1,filenam2,filenam3));
filenam5=char(string('images-'));
dirfile=char(strcat(filenam5,filenam1));
fullFileName = fullfile(dirfile,filenam4);
A=imread(fullFileName);
% Passar a escala de grisos i binaritzar la imatge;
Agray=rgb2gray(A);
Abin=Agray>128;
[altoimagen,amplada]=size(Abin); %Capturem les mides de la imatge original
numdigits=0;    % En aquest comptador tindrem el valor de numero d'imatges trobades
colini=zeros(10); %Guardarem aquí la posició inicial de cada dígit trobat
colfin=zeros(10); %Guardarem aquí la posició final de cada dígit trobat
imatgetrobada=0;

% Recorregut per analitzar totes les columnes
for amp=1:amplada
    if Abin(:,amp)==1                 %Si trobem una fila buida
        if imatgetrobada==1           % i estem recorrent un dígit,
          imatgetrobada=0;            % netegem indicador de dígit trobat
          colfin(numdigits)=amp-1;    % i guardem la posició anterior com a final d'un dígit
        end
    else                              % Si trobem una fila no buida
        if imatgetrobada==0           % Si a la posició anterior no havia imatge
          imatgetrobada=1;            % Activem indicador de dígit trobat
          numdigits=numdigits+1;      % incrementem comptador de dígits trobats
          colini(numdigits)=amp;      % i guardem la posició com a inici d'un dígit
        end
    end
end
% Recorregut per retallar els dígits i gravar-los en arxius separats
for dig=1:numdigits
        final=colfin(dig)-colini(dig)+1;
        digito=imcrop(Abin,[colini(dig),1,final,altoimagen]);
        [altodigito,ampladadigito]=size(digito);
        if altodigito>ampladadigito
                digito2=ones(altodigito,altodigito);
                diferencia=round((altodigito-ampladadigito)/2);
                amplada2=ampladadigito+diferencia-1;
                indice=1;
                for despl=diferencia:amplada2
                        digito2(:,despl)=digito(:,indice);
                        indice=indice+1;
                end
```

```
        else
                digito2=digito;
        end
        % Conversió d'imatge binària a RGB
        grayImage = 255 * uint8(digito2);
        RGB = cat(3, grayImage, grayImage, grayImage);

        % Codi per preparar el nombre de l'arxiu i gravar-lo
        filenam1=char(string(Numinp));
        filenam2=char(string(dig));
        filenam3=char(string('.png'));
        filenam4=char(strcat(filenam1,filenam2,filenam3));
        filenam5=char(string('images-'));
        dirfile=char(strcat(filenam5,filenam1));
        fullFileName = fullfile(dirfile,filenam4);
        imwrite(RGB,fullFileName);
end
```